НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ФІЗИКО-МЕХАНІЧНИЙ ІНСТИТУТ ім. Г. В. КАРПЕНКА

На правах рукописи

ФАРИД Талаат Мохамед

ОПТИМИЗАЦИЯ ИСПОЛЬЗОВАНИЯ ПАМЯТИ В ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ СИСТЕМАХ ОБРАБОТКИ ИНФОРМАЦИИ

Специальность 05.13.04 - автоматизированные системы управления и системы обработки информации

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата технических наук

Львів — 1995

Диссертация является рукописью.

Работа выполнена в Белорусском государственном университете Министерства образования и науки Республики Беларусь.

Научные руководители: доктор физико-математических наук Вальковский Владимир Александрович (г.Львов, Украина), кандидат технических наук, доцент Курбацкий Александр Николаевич (г.Минск, Беларусь)

Официальные оппоненты: доктор физико-математических наук, профессор Кожевникова Галина Павловна, кандидат технических наук, старший научный сотрудник Деркач Богдан Теодорович

Ведущая организация: Институт кибернетики им. В.М.Глушкова НАН Украины

Защита состоится "04" "07" 1995 г. в 15 часов на заседании Специализированного Ученого Совета Д. 04.01.02 при Физико-механическом институте НАН Украины (290601, Львов, ул. Научная, 5)

С диссертацией можно ознакомиться в библиотеке института.

Автореферат разослан "02" "06" 1995 г.

Ученый секретарь
специализированного ученого Совета
к.т.н., ст.н.с. Р.А.Бунь

Общая характеристика работы

Актуальность темы. Проблема распараллеливания и конвейеризации вычислений является весьма актуальной по нескольким причинам. Главные из них: быстро развивающаяся архитектура вычислительных систем, за которой часто не успевает программное обеспечение; наличие большого количества традиционно написанных последовательных программ; отсутствие необходимых вычислительных мощностей, особенно в странах третьего мира. Данная работа посвящена проблемам максимального использования вычислительных ресурсов при минимальных затратах на них. Главным образом, решается задача совместной оптимизации программ по памяти и увеличения их потенциального параллелизма, что существенно степень исследовательности. Это тем более актуально, что баланс экономики стран третьего мира складывается не в их пользу. Основные ресурсы лежат на пути использования передовых технологий, в том числе информационных. Известно, что задача оптимизации машинного кода для конвейерных процессов сводится к задаче минимизации длины расписания выполнения заданий, связанных ориентированным графом зависимости между операторами. При этом дуги графа помечены весами - некоторыми целыми значениями, зависящими от глубины конвейера для соответствующей вершины-операции, из которой исходит дуга. Решая проблему оптимизаций машинного кода, исследователи исходят, как правило, из этого представления, используя все более тонкие эвристики и все более сложные субоптимальные алгоритмы. Но совсем немного работ, которые бы делали попытку вернуться на шаг назад - проанализировать исходную машинную программу и может быть, путем нетрудоемких ее преобразований типа переименования регистров (переменных), перестановки соседних операторов, снятия

старых или введения новых ограничений - сделать образующийся из нее далее ориентированный граф более предпочтительным для целей последующей упаковки. В данной работе предлагается именно такой путь. Исходная программа подвергается разного рода воздействиям для того, чтобы результирующая программа была оптимальна: либо по времени; либо по числу используемых быстрых регистров; либо по числу обращений к медленной памяти. Поэтому актуальность данной работы не вызывает сомнений.

Цель работы и задачи исследования. Целью работы являлось комплексное исследование проблемы оптимального использования памяти в контексте проблемы конвейеризации вычислений, а также смежных с этой проблемой вопросов. В постановку задачи входило - привнести в традиционные методы составления расписаний заданий для ЭВМ ряд элементов асинхронного программирования, которое весьма эффективно развивается в странах СНГ. Главная цель - разработать методы оптимизации программ по двум противоречивым критериям. Первый критерий - минимизация числа используемых регистров и ячеек памяти. Второй - максимальное снятие зависимостей между операторами с тем, чтобы последующее применение алгоритмов оптимизации программ для конвейерных вычислителей давало большие возможности для упаковки. При этом ставилась побочная цель: использовать методы асинхронизации для качественного улучшения этих алгоритмов. В данном подходе удалось этого достичь. Цель работы достигнута благодаря известным методам распараллеливания, развитым в работах моих научных руководителей.

Методы исследования. В работе употреблялся весь спектр методов исследования дискретной математики. Наиболее активно использовались теория графов, в частности, элементы раскраски графов; а также теория расписаний и теория программирования.

Многие из используемых методов основаны на теории оптимизации. В работе использованы также некоторые идеи автора, накопленные во время работы над диссертацией магистра наук [3].

Научная новизна работы заключается в следующем: впервые удалось соединить воедино приемы асинхронизации и приемы составления оптимальных расписаний. До этого эти два направления существовали, как бы, по отдельности. Применение приемов асинхронизации позволило достичь качественно нового уровня составления расписаний для прохождения вычислительных работ. В частности, автором предложен оптимальный алгоритм экономии регистров без допущения перестановочности операторов. Им же также разработано семейство аналогичных субоптимальных алгоритмов, исследована проблема их эффективного программирования на языке высокого уровня, а также проблема конвейеризации циклов.

Практическая ценность работы заключается в том, что разработанные в ней методы и алгоритмы могут быть использованы в оптимизирующих трансляторах практически для любой ЭВМ. Особенно это касается конвейерных вычислителей RISC-архитектуры с небольшим объемом регистровой памяти. Об этом есть необходимые справки.

На защиту выносятся следующие результаты.

1. Комплекс формальных алгоритмов различной степени параллелизма и различной ориентации для экономии быстрых регистров спецпроцессоров архитектуры типа RISC.

2. Аналогичный комплекс для целей более эффективного использования памяти, в частности, для экономии обращений к

памяти, одновременно с оптимальным использованием быстрых регистров.

3. Совокупность методов и алгоритмов для элиминации связей между операторами машинного кода в целях последующей оптимальной его упаковки. Отметим, что эти алгоритмы присутствуют в работе "в контексте" алгоритмов, описанных выше, т.е. выполняются как отдельные шаги. Но они могут быть использованы и вполне самостоятельно для снятия непринципиальных зависимостей между операторами.

4. Представление описанных алгоритмов на языке близком к реальному языку программирования высокого уровня. Изучены вопросы оптимальной их реализации, включая структуры данных.

Реализация результатов работы. Основные результаты работы были реализованы в оптимизирующих трансляторах для спецпроцессоров в рамках хоздоговорных и государственных научно-исследовательских тем. В частности, они были использованы в НПО "КВАНТ" (г.Москва) при выполнении работ "Исследование и разработка алгоритмов машинно-зависимой оптимизации программ для ЭВМ с конвейерной архитектурой" (Д8676/10/22) и "Исследование и разработка алгоритмов оптимизации при реализации трансляторов с языков программирования для ЭВМ с конвейерной архитектурой" (Д8676/10/41-91); в НИР "Автоматизация распараллеливания и векторизации программ в ориентации на многопроцессорные вычислительные комплексы и транспьютерные сети", выполненной по программе "Перспективные информационные технологии и системы" (ГКНТ Украины). Они также были использованы в ФМИ НАН Украины и НИИ ЭВМ (г.Минск).

Апробация работы. Работа докладывалась на международной научно-технической конференции "Информационные технологии и системы" - ITIC - 93 (Львов), симпозиуме "Вопросы оптимизации вычислений" (Киев, 1993), конференциях молодых ученых Белгосуниверситета (Минск, 1993, 1994), а также на научных семинарах ФМИ НАН Украины (Львов), кафедры технологии программирования Белгосуниверситета (Минск).

Публикации. По теме диссертационной работы опубликовано 5 научных работ. Личный вклад автора состоит в следующем. В работе [1] выполнена обзорная часть по вопросам конвейеризации и векторизации. В [2] - разработаны основные алгоритмы экономии регистровой памяти. В работах [4,5] авторы приняли равное творческое участие.

Структура и объем работы. Диссертация состоит из введения, пяти глав, выводов, списка литературы и приложений. Она изложена на страницах, включая рисунков и список литературы.

Содержание работы

Во введении обоснована актуальность темы исследований, сформулирована цель работы, научная новизна и практическая ценность результатов, дана краткая аннотация разделов диссертационной работы.

Первая глава посвящена анализу проблемы оптимизации программ в общей постановке. В ней предлагается оригинальная классификация методов оптимизации вообще, даются многочисленные примеры, а далее показано, какое место занимают предлагаемые в работе методы в общем контексте проблемы оптимизации.

Классификация имеет четыре основных ключа.

1. Источник оптимизации: указание на то, за счет чего осуществляется оптимизация. Предлагается рассмотреть три основных источника: внутримодульная экономия вычислений на регистрах за счет их рационального использования; оптимизация взаимодействия регистров с внешней памятью; уменьшение числа несанкционированных прерываний.

2. Оптимизируемые языковые конструкции: линейные участки программ; ациклические условные структуры; циклы; процедуры.

3. Уровень оптимизации: машинный язык; ассемблер с логическими, т.е. с нераспределенными по памяти и по регистрам данными; ассемблер с укрупненными операциями (сложение, умножение, деление...); ассемблер, сочетающий два предыдущих; язык высокого уровня.

4. Способ оптимизации: машиннонезависимая экономия числа операций; экономия числа команд за счет совмещения операций в команде; экономия числа операций за счет усложнения операций; упреждающая подкачка; использование эффекта "гребенки"; уменьшение накладных расходов.

Классификация относится к методике оптимизации только в рамках одного модуля. При наличии нескольких модулей приведенный перечень значительно расширяется за счет возможностей межмодульного распараллеливания, планирования и оптимизации межмодульных взаимодействий. В главе подробно рассмотрены такие оптимизационные приемы, как разгрузка участков повторяемости; распроцедуривание; препроцессинг на этапе трансляции; чистка программы; экономия памяти; расчленение циклов; уменьшение степени вложенности циклов, а также весьма подробно описаны

методы векторизации разных конструкций. Заканчивается глава рассмотрением методов оптимизации на уровне машинного кода.

Во второй главе предложены методы и алгоритмы экономии регистров и обращений к памяти. Проблема оптимизации программы по памяти заключается в преобразовании ее в эквивалентную, имеющую минимальное число регистров и (если регистров не достаточно) осуществляющую минимальное число обращений к внешней памяти. Уточнение этой постановки приводит к рассмотрению целого спектра методов и алгоритмов оптимизации.

Перечислим основные из них.

1. Алгоритм A1 в применении к программе S "освобождает" ее от всех конкуренционных связей, присутствующих в S "сверх" информационных связей и дает минимальное число ячеек памяти в результирующей программе среди всех таких (т.е. ликвидирующих сверхнормативные конкуренционные связи) алгоритмов.

2. Алгоритм A2 отображает логическую память в физическую адекватно, т.е. разным ячейкам сопоставляет разные, одинаковым - одинаковые.

3. Алгоритм A3 сохраняет, не увеличивая, общую (т.е. объединение информационной и конкуренционной) зависимость и выдает результирующую программу, имеющую минимальное число ячеек.

4. Алгоритм A4 преобразует программу S в программу с минимальным числом ячеек, не взирая на увеличение при этом конкуренционных связей.

Дополнительно к любому из четырех требований на алгоритм может быть добавлено специальное условие B: при ограниченном

числе регистров M в случае, если регистров для организации правильного обмена информацией не достаточно, нужно минимизировать число обращений к тем ячейкам, число которых выходит за пределы M, т.е. ячейкам внешней памяти. Имеет место

Лемма. Если Ai(S) - граф общей зависимости (объединение информационной и конкуренционной зависимости) результата применения алгоритма Ai к программе S, Ri(S) - число используемых в итоге регистров , то

а) $A1(S) \le A2(S) = A3(S) \le A4(S)$;

б) $R1(S) \ge R3(S) \ge R4(S)$.

Среди перечисленных алгоритмов Ai интерес представляют A1 и A4. Первый рекомендуется применять при достаточно "редкой " сети фактических связей в программе. Второй, например, в случае тотальной связи между операторами (каждый последующий зависит от предыдущего), когда появление новых конкуренционных зависимостей уже ничего не может "испортить".

Опишем основные шаги алгоритма A4.

Шаг_1. Построение информационного графа "на полюсах " (P(S)) и информационного графа на операторах S.

Шаг_2. Построение таблицы ТИР(S) - использования регистров (ячеек). Если r - число используемых программой S регистров, k - число ее операторов, то ТИР(S) имеет размер $r \times (k-1)$, в позиции (i,j) содержит 1, если i-й регистр при переходе от j-го к j+1-му оператору содержит используемую далее информацию. Таблица порождает массивы Z[1:k-1]: Z(i)- это число занятых (по существу) в момент i регистров; Ch[1:r]:Ch(i)- число записей и считываний i-го регистра.

Шаг_3. Находится $M=\max\{z(i):i=1,...,k-1\}$.

Шаг_4. Заключается в применении некоторой процедуры, осуществляющей сокращение числа регистров программы S до величины M.

Лемма. Описанный алгоритм преобразует программу S в эквивалентную по информационному графу и имеющую минимальное число регистров среди всех эквивалентных S программ.

В главе предложен также формальный язык для описания процесса преобразования путем алгоритмов A1 - A4.

Общая стратегия решения задачи при выполнении условия B включает следующие шаги:

1. Применение алгоритма A4. Если число используемых в S ячеек не очень велико, то этот шаг может быть опущен. Наиболее "тщательная" оптимизация должна включать обе возможности: параллельное применение стратегии с предварительной обработкой алгоритмом A4, и без нее.

2. Выбор ячеек - кандидатов на регистры. Он осуществляется с использованием одного из критериев "популярности" ячейки.

3. Вложение рабочих интервалов оставшихся ячеек, т.е. кандидатов на ячейки внешней памяти, в пустые интервалы ячеек-регистров. Простейшая стратегия вложения - от пустых интервалов ячеек-регистров, начиная с самых длинных, к рабочим интервалам ячеек памяти.

4. Замена длинных связей через ячейки-регистры на серии коротких связей через внешнюю память. Используемые стратегии такие же, как и в предыдущем пункте.

Результаты главы иллюстрируются несколькими примерами.

Третья глава посвящена алгоритму A1, ликвидирующему конкуренционные связи, и его модификациям. Алгоритм включает следующие основные шаги.

Шаг_1. Строятся графы P(S) и S. Это в точности первый шаг алгоритма, описанного в предыдущей главе.

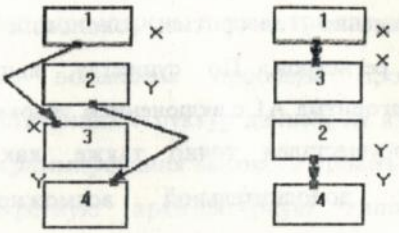
Шаг_2. Проверка связности графа S. Если граф S не является связным, то программа S разбивается на соответствующее число участков, и для каждого из них задача решается в отдельности.

Шаг_3. Строится граф зацепленности C(S)-компонент графа P(S). Как обычно, два компонента P1 и P2 графа P(S) являются зацепленными, если есть два оператора $f1 \in P1$ и $f2 \in P2$, не находящиеся в информационной зависимости и попадающие в конкуренционную зависимость в случае, если компонентам P1 и P2 будет приписана одна и та же переменная.

Шаг_4. Минимальная раскраска графа C(S).

Шаг_5. Переименование ячеек памяти. Компоненты связности графа P(S) получают новые имена - результат раскраски.

Дальнейшая экономия времени использования регистров связана с его уменьшением путем перестановки операторов. Тривиальный пример такой перестановки даст преобразование:



В первой схеме регистры X и Y заняты по два такта, во второй - по одному. Опишем вначале технику экономии за счет локальных перестановок. Определим так называемый вектор экономии **e**(S). Он будет содержать n-1 компонент. Пусть S(j,k) - схема, полученная из S перестановкой j-го и k-го операторов. T(S) - чистое время использования регистров в схеме S. Тогда i-й компонент пуст, если i-й и i+1-й операторы находятся в зависимости, т.е. их перестановка не допустима. В противном случае i-ый компонент равен числу T(S(i,i+1)) - T(S) т.е. количеству тактов занятости регистров, сэкономленному благодаря перестановке i-го и i+1-го операторов. Все описанное может быть выражено как очередной шаг алгоритма.

Шаг_6. Последовательное применение процедуры локальной экономии времени чистого использования регистров до тех пор, пока вектор **e** содержит положительные компоненты. Ясно, что описанный шаг даст некоторый локальный минимум функции T(S), но, вообще говоря, не дает глобального. Для получения глобального минимума нужно проверить все перестановки между участками операторов, причем, не только соседними. Для больших программ это весьма трудоемкая процедура. Она может быть рекомендована при $n \le 10$.

Шаг_7. Применение алгоритма экономии общего времени использования регистров. По существу, является переходом к модификации алгоритма A1 с включением условия B. Учет условия B может быть осуществлен точно также, как и в главе 2, с использованием дополнительной возможности перестановки регистров.

Но в работе рекомендована также и другая техника. Опишем ее суть.

Если мы хотим как можно больше интенсифицировать использование какого-то регистра, например X, для передачи информации, нужно покрасить цветом X как можно больше вершин графа C(S). Возможны следующие варианты при окраске двух вершин одной краской.

1. Вершины смежны, их окраска одним цветом допустима и не ведет к увеличению числа дополнительных, т.е. конкуренционных связей в схеме S.

2. Вершины несмежны, однако соответствующие компоненты "не зацепляются" (номера всех операторов одного компонента меньше, чем другого), а следовательно окраска одним цветом возможна, но приводит к появлению дополнительных конкуренционных связей.

3. Вершины несмежны, соответствующие компоненты "зацепляются", но есть допустимая перестановка операторов, после которой они оказываются "незацепленными". После перестановки возможна окраска одним цветом с появлением дополнительных конкуренционных связей.

4. Вершины несмежны, "зацепляются" и никакая допустимая перестановка их "не расцепляет".

В работе подробно рассмотрены первые три случая.

Четвертая глава посвящена проблеме программирования алгоритмов и соответствующих структур данных на языке, близком к реальным языкам программирования высокого уровня и в ориентации на некоторую конкретную архитектурную гипотезу. Вначале рассмотрена проблема оптимизации программ взаимодействующих с памятью. Во всех предыдущих рассмотрениях считалось, что операторы программ взаимодействуют над логическими регистрами. Допущения взаимодействия над памятью усложняют наши методы. Информационная и конкуренционная зависимости выявляются сложнее и могут зависеть от не всегда предсказуемых условий и величин. Рассмотрим несколько подходов, позволяющих решить поставленные задачи при взаимодействиях операторов над памятью. Решение достигается некоторым "огрублением" используемых конструкций и дальнейшим сведением постановок задач к уже рассмотренным постановкам.

Итак, пусть в дополнение к обычным именам логических регистров полюса, т.е. входы и выходы операторов, могут помечаться обращениями к памяти вида: {A}, {A±1},{A±C}, {A+B},{A+B±1},{A+B±C}, где A - адресный или общий регистр, B - регистр смещения, C - целая константа.

Если в программе два полюса помечаются обращениями к памяти {t} и {τ}, то в случае {t}={τ} через них может осуществляться информационная связь или возникать конкуренция над памятью. Некоторые рекомендации в работе даны и для этого случая.

Следующий раздел посвящен особенностям программирования алгоритмов оптимизации.

Начнем с алгоритма А4. Как уже было замечено, таблица использования регистров может быть представлена матрицей, строки которой соответствуют тактам, а столбцы- логическим регистрам: на позиции (i,j) стоит 1, если j-й регистр на i-ом такте используется для передачи информации.

Алгоритм экономии числа регистров А4 реализует следующая программа.

1: $M \leftarrow \max \{ \sum_{r=1}^{m} T(i,r): i=1,...,n\}$;
2: $i \leftarrow 0$;
3: $S \leftarrow \varnothing$;
4: пока $|S| \le M \wedge i \le n$ выполнять
5: [$i \leftarrow i+1$;
6: $R \leftarrow \{ l : l \notin S \wedge T(i,l) = 1\}$;
7: $S \leftarrow S \cup R$];
8: если $i = n$ то конец вычисления;
9: для произвольного $q \in R$ выполнять
10: [$j \leftarrow 1$;
11: пока $T(i+j, q) = 1$
12: выполнять $j \leftarrow j+1$];
13: для всех $u \in S \backslash R$ выполнять
14: [$s(u) \leftarrow 0$;
15: пока $T(i+s(u),u) = 0$
16: выполнять $s(u) \leftarrow s(u)+1$];
17: если $v \in S \backslash R$: $s(v) \ge j$;
18: то
19: для всех $h \in [0, j-1]$ выполнять;
20: [$T(i+h,v) \leftarrow 1$;
21: $T(i+h,q) \leftarrow 0$] ;
22: $S \leftarrow S \backslash \{q\}$; $R \leftarrow R \backslash \{q\}$;

23: на 4;

24: иначе для произвольного w: T(i,w) = 0 ∧ w ∈ S\R

25: выполнять

26: для всех h ∈ [1, i-1] выполнять

27: [T(h,q) ← T(h,w);

28: T(h,w) ← 0]

29: S ← S\{w}; R ← R\{q};

30: на 4.

Дадим к ней необходимые комментарии.

Оператор 1 вычисляет число M - минимальное возможное число используемых регистров. Цикл 4 осуществляет просмотр тактов с формированием множества R, вновь появившихся используемых регистров, множества S - общего множества использованных до данного такта регистров, и с проверкой превышения мощности множества S выше "критического" числа M. Далее, по условию окончания разбора стоит выход из программы. Операторы 10-12 вычисляют длину интервала из единиц, т.е. времени использования регистра q, который выводит множество S за пределы допустимого числа M. Операторы 14-16 параллельного цикла 13 делают почти то же самое со всеми "старыми" используемыми регистрами - вычисляют длину интервалов из нулей, на место которых может быть помещен интервал из единиц регистра q. Условие 17 проверяет эту возможность. Если требуемый регистр v находится, то операторами 20 и 21 на место интервала из нулей регистра v переносится интервал из единиц регистра q. Соответственно модифицируются множества S и R, и потактовый анализ повторяется. Если же среди старых регистров не находится ни одного "вмещающего" интервала использования регистра q, то для некоторого из них, не используемого на данном такте, вся история его использования от 1-го до i-1-го такта

переписывается на место регистра q. После модификации S и R анализ повторяется.

Ниже изображена программа, осуществляющая уплотнение одного выбранного регистра активными интервалами других регистров. Второй оператор программы среди всех регистров выбирает такой регистр k, для которого сумма элементов столбца матрицы T(i,k) максимальна. Операторы 3 и 4 формируют для k-го регистра матрицу I размеров r×2. Она "склеивается" из векторов $(j_1,...,j_r)$ и $(l_1,...,l_r)$. Числа j_s показывают начала интервалов из нулей, т.е. начала "дырок" в истории использования регистра k. Числа l_s выражают длины этих "дырок". Для всех других регистров, кроме k также проводится одновременный анализ их историй. Это осуществляется операторами 6 и7, которые очень похожи на 3 и 4. Разница только в том, что в оставшихся регистрах ищутся не "дырки", а наоборот, интервалы из единиц. Числа j_s^t показывают начала таких интервалов для t-го регистра, а l_s^t- их длины. Собранная информация хранится матрицами I_t.

1: если m < 2 то конец вычисления;

2: $k \leftarrow p$: $\sum_{i=1}^{n} T(i,p) = \max \{ \sum_{i=1}^{n} T(i,j),\ j=1,...,m\}$;

3: $I(\ ,1) \leftarrow (j_1,...,j_r)$: $\forall s\ (T(j_s,k) = 0 \wedge T(j_s-1,k) \neq 0)$;

4: $I(\ ,2) \leftarrow (l_1,...,l_r)$: $\forall s\ (\bigwedge_{i=0}^{l_s-} T(j_s+i,k) = 0) \wedge T(j_s+l_s,k) \neq 0)$;

5: для всех $t \neq k$ выполнять

6: $[I_t(\ ,1) \leftarrow (j^t,...,j^t_{r_t})$: $\forall s\ (T(j_s,t) \neq 0 \wedge T(j_s-1,t) = 0)$;

7: $I_t(\ ,2) \leftarrow (l^t,...,l^t_{r_t})$: $\forall s$ конъюнкция $T(j^t_s+i,t) \neq 0) \wedge T(j^t_s+l^t_s,t) = 0)]$;

8: для всех i = 1,...,r; j = T(i,1); l = T(i,2) выполнять

9: если $\exists t,q,\ j^t = l_t(q,1),\ l^t = l_t(q,2)\ (j \le j^t \wedge j+l \ge j^t+l^t)$

10: то для всех $p = j^t, j^t+1, \ldots, j^t+l^t$ выполнять

[$T(p,k) \leftarrow T(p,t);\ T(p,t) \leftarrow 0$];

11: $T \leftarrow T\setminus\{ T_k \cup \{ T_j : \sum_{i=1}^{n} T(j,i) = 0\}\}$;

12: $m \leftarrow |\{T_k: T_k \in T\}|$;

13: на 1;

Аналогичным образом в главе описаны остальные алгоритмы.

Наиболее глобальный подход при экономии регистров заключается в рассмотрении всех используемых регистрами интервалов безотносительно их привязки к конкретным именам регистров. Каждый интервал характеризуется началом j^t, длиной l^t, величиной b^t и относительной интенсивностью его использования для записи и считывания $c^t = b^t / l^t$. При применении данного подхода мы стараемся "набить" истории регистров интервалами с наибольшими значениями величин c^t. Как видим, задача сводится к одной из модификаций задачи об упаковке ранцев. Если определить понятие совместимости интервала t со множеством интервалов S как непересекаемость t с каждым интервалом, входящим в S, обозначить через D-множества всех интервалов, то простейшая процедура упаковки одного регистра выглядит так:

1: $S \leftarrow \varnothing$

2: $S \leftarrow S \cup \{t\}: t \in D$ и c^t -наибольшее среди всех $t \in D$;

3: $D \leftarrow$ {t:t совместим с S}

4: На 2

В конце главы делается вывод, что трудоемкость реализации основных алгоритмов оптимизации не выше $O(n^3)$.

Пятая глава - состоит из трех параграфов, в которых рассмотрены важные смежные вопросы.

Вначале приводится некоторая архитектурная гипотеза, имеющая под собой реальный прототип - процессор транспьютерного типа, разработанный в одном из институтов г. Москвы. Дальнейшее изложение сориентировано на этот тип архитектуры.

В главе также рассмотрена проблема конвейеризации циклов. Под конвейеризацией витков циклов, как обычно, понимается совмещение во времени выполнения соседних итераций цикла. Конвейеризация начинается с изучения структуры информационной конкуренционной зависимости. Ее можно разделить на внутриитерационную и межитерационную. Графы внутриитерационной зависимости строятся обычным образом. Межитерационная зависимость связывает операторы различных итераций и требует особого исследования. Остановимся на этом подробнее.

Поскольку у нас идет речь о совмещении соседних итераций, каждой последующей с предыдущей, то любая межитерационная зависимость, связывающая операторы i-й и j-й итерации при $j-i \geq 2$, при глубине конвейера 2 будет сохраняться при любом таком совмещении. Поэтому нас будет интересовать связь только между операторами соседних итераций. На этот счет доказана лемма.

Лемма. Если считать, что любая пара обращений к памяти порождает потенциальную связь, то для любых i,j информационные и конкуренционные графы, построенные на парах итераций i,i+1 и j,j+1 совпадают.

В общих чертах, техника совмещения заключается в следующем.

1. Для 1-й и 2-й итерации строится граф информационной зависимости G.

2. Осуществляется "разрядка" итераций, т.е. каждому оператору "приписывается" нечетное место: 1,3,5,...

3. Вторая итерация "вставляется" в четные места первой итерации на позиции 2,4,6,...

4. Строится информационный граф G' для полученной конструкции.

5. Если G'=G , то попытка удалась, и совмещение возможно, иначе

6. Вторая итерация сдвигается на одну позицию, занимая места 4,6,8,..., для полученной конструкции строится граф G" с графом G. Если G"=G, то попытка удалась, иначе осуществляется сдвиг на позиции 6,8,10,... Процесс повторяется до полного вырождения совмещения итераций.

Это, и другие тонкости, связанные, в частности, с конкретной архитектурной гипотезой, также рассмотрены в главе.

Основные результаты работы

Если говорить о содержании работы в целом, резюмируя все выше изложенное, то в ней

1. Предлагается и обосновывается многопараметрическая классификация методов оптимизации алгоритмов и программ. Основные результаты работы занимают вполне определенные позиции в рамках этой классификации.

2. Разработаны оптимальные методы и алгоритмы экономии быстрых регистров и обращений к памяти. Используемая техника не затрагивает порядок выполнения операторов и информационно-логическую структуру программы, а основана только на переименовании регистров и интенсификации их использования.

3. Разработаны субоптимальные методы и алгоритмы для тех же целей, использующие прием снятия непринципиальных связей между операторами, их перестановки и методику раскраски вершин информационного графа.

4. Основные алгоритмы записаны в языках, близких к языкам программирования высокого уровня.

5. Разработана универсальная стратегия их использования для целей оптимизации конвейерных вычислений, в частности, в ориентации на некоторую архитектуру специального типа.

Публикации по теме диссертации

1. Вальковский В.А., Костовский В.А., Курбацкий А.Н., Фарид Т.М. Некоторые вопросы оптимизации программ. - Львов, 1994. - Препринт / АН Украины. Ин-т прикладных проблем механики и математики им. Я.С.Подстригача; №3 - 94 - 19 с..

2. Вальковский В.А., Курбацкий А.Н., Фарид Т.М. Алгоритмы оптимизации программ для конвейерных ЭВМ. - Львов, 1994. - Препринт / АН Украины. Ин-т прикладных проблем механики и математики им. Я.С.Подстригача; №6 - 94 - 23 с.

3. Farid T.M. Performance evaluation of the Combined CSMA/CD and hub polling protocols in the LAN using simulation// Ms.D. Thesis, Suez Canal University, Egypt, 1988. - 98 p.

4. Farid T.M., Valkovskij V.A. Parallelization and pipeline implementation of program loops// Міжнародна конференція з інформаційних технологій і систем, 1993, Львів. - С. 30-33.

5. Вальковский В.А., Заярская Е.В., Фарид Т.М. Параллельно-конвейерные методы для решения одного класса задач обработки

сигналов// Тези доповідей симпозіуму "Питання оптимізації обчислень", Київ, 1993. - С.37.

6. Валькевский В., Фарид Т.М. Экономия памяти: Использование техники асинхронизации. Всеукраинская конференция "Розробка та застосування математичних методів в науково-технічних дослідженнях". Львів, 1995. (в печати)

7. Куроацкий А.Н., Фарид Т.М. О связи между разработкой логических систем и распараллеливания циклов. Вторая украинская конференция по автоматическому управлению, Львов, 1995. (в печати).

8. Farid T.M. Parallel Processing in Computer Nets and Transputer Nets: Mutual ideas transporting// Second ukrainian conference on automatic control "Automatics - 95", Lviv, 1995 (в печати).

Фарид Т.М. Оптимизация использования памяти в высокопроизводительных системах обработки информации. Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.04 - автоматизированные системы управления и системы обработки информации, Белорусский государственный университет, Минск, 1995.

Исследована проблема оптимального использования регистровой и оперативной памяти в конвейерных вычислителях. Предложены алгоритмы повышения потенциального параллелизма программ путем снятия непринципиальных зависимостей. Разработан метод конвейеризации циклов. Предложена стратегия комплексного использования введенных методов в контексте общей проблемы составления расписаний для компьютерных программ.

Ключевые слова: оптимизация программ, распараллеливание и конвейеризация вычислений, регистровая память, упаковка машинного кода, система обработки информации.

Farid T.M. Optimization of using of memory in high-speed information processing systems. Dissertation for obtaining of scientific degree of candidate of science (engineering) on speciality 05.13.04 - automatized control systems and information processing systems. Byelorassian State University. Ministry of Education and Science of Republic of Belarus, Minsk, 1995.

The problems of optimal using of register and read/write memory in pipe-line computers is investigated. The algorithms of increase of means of elimination of inessential connections are proposed. A method for pipelining of loops is developed. A strategy for coordinated application of the methods introduced in context of general scheduling problem for computer programs is proposed.

Key words: program optimization, parallelization and pipelining of computations, register memory, machine code packing, information processing system.

Т. Фарид

Подп. в печать Формат 60x84/16. Бумага тип. № 2.
Печать офсетная. Усл. печ. л. 1.31 Усл.кр.-отт. л. - 1.31
Тираж 100. Зак. 1566

Белгосуниверситет Министерства образования и науки Республики Беларусь, 200080, Минск, п. Скорины 4.

Ключевые слова: оптимизация программ, распараллеливание и конвейеризация вычислений, регистровая память, упаковка машинного кода, система обработки информации.

Farid T.M. Optimization of using of memory in high-speed information processing systems. Dissertation for obtaining of scientific degree of candidate of science (engineering) on speciality 05.13.04 - automatized control systems and information processing systems. Byelorussian State University, Ministry of Education and Science of Republic of Belarus, Minsk, 1999.

The problems of optimal using of register and associative memory in long-line computers is investigated. The algorithms of decrease of volume of elimination of inessential connections are proposed. A method of optimizing of loops is developed. A strategy for coordinated application of the methods introduced in context of general scheduling problem for computer programs is proposed.

Key words: program optimization, parallelization and pipelining of computations, register memory, machine code packing, information processing system.