УДК 004.272.2, 004.272.23

В.А. Клименко
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

ПЕРЕТВОРЕННЯ ТІЛА ВНУТРІШНЬОГО ЦИКЛУ У ВЕКТОРНУ ІНСТРУКЦІЮ ДЛЯ КОНФІГУРОВАНОЇ СИСТЕМИ

Розглянуто новий підхід до побудови векторних інструкцій. Визначено основні недоліки та переваги використання векторних інструкцій для підвищення продуктивності виконання програми. Запропоновано використання спеціальної векторної інструкції для конфігурованої системи. Визначено основні обмеження застосування цього підходу. Наведено алгоритм перетворення внутрішнього тіла циклу в спеціалізовану векторну інструкцію.

Ключові слова: векторна інструкція, конфігурована система.

A new approach to create vector instruction was investigated in this paper. Advantages and disadvantages of vector instruction using for program acceleration were considered. Using special vector instruction for configurable computer system was proposed. All constrains for transformation loop body in special vector instruction were defined. Algorithm for transformation inner loop body in special vector instruction was introduced in this paper.

Key words: vector instruction, configurable system.

Вступ

Використання векторних інструкцій дозволяє істотно збільшити продуктивність системи під час роботи з однорідними даними. Застосовуючи векторні інструкції у конфігуроваиій системі, набір векторних інструкцій не є фіксованим, що дозволяє гнучкіше використати переваги таких інструкцій. Використання можливостей конфігурування апаратури під задачу дозволяє створювати спеціалізовані векторні інструкції при компіляції програми та конфігурування апаратури відповідно до створеної системи векторних команд. Здебільшого основний час виконання програми витрачається у найбільш вкладених циклах. Оптимізація з метою прискорення виконання вкладених циклів дає змогу отримати істотний ефект для всієї програми. Векторною інструкцією вважатимемо інструкцію, яка має множину вхідних та вихідних даних та множину елементарних операцій, з якої складається векторна операція. Конфігурованою системою вважатимемо систему, набір та формат команд якої формується відповідно до вхідного алгоритму програми. При компіляції програми формується бінарний файл виконавчої програми та опис апаратних засобів, на яких буде виконуватися ця програма.

Огляд літературних джерел

Векторні інструкції знаходять своє застосування у векторних комп’ютерах для розв’язання складних наукових задач. Характерною ознакою для таких задач є велика кількість однорідних обчислень над великими масивами даних. В універсальних комп’ютерах векторні інструкції знайшли своє застосування для оброблення мультимедійних задач на графічних процесорах. Останнім часом поширеною стає задача застосування графічних процесорів та векторних інструкцій для прискорення програм персонального комп’ютера. Прискорювачем універсального комп’ютера може виступати не тільки графічний сопроцесор, але й спеціалізоване конфігуроване ядро, яке разом з універсальним процесором утворюють конфігуровану систему. Виконання векторних інструкцій на такій системі дозволить отримати підвищення продуктивності та вищу ефективність застосування векторних інструкції. Оскільки більшість програм описується послі-

довними мовами, створення векторної програми в тому чи іншому ступені покладається на компілятор. Постає задача створення алгоритму виділення та утворення векторних інструкції для конфігурованих систем.

Класифікація типів оптимізуючих систем за рівнем оптимізації [1].

1. Автоматичний – компілятор сам вирішує, які частину коду оптимізовувати.
2. Автоматизований – користувач вказує компілятору, які частини та які види оптимізації застосовувати:

а) використання оптимізованих бібліотек під конкретну архітектуру;

б) використання директив.

3. Ручна оптимізація – користувач самостійно оптимізує програму, використовуючи спеціалізовані інструкції, змінюючи алгоритм відповідно до архітектури комп'ютера.
4. Комбінована оптимізація – користувач застосовує комбінації вищезгаданих типів.

У цій роботі розглядаються підходи та рішення для першого виду системи, оскільки така система дозволяє зменшити витрати на розроблення. Загальні обмеження універсального алгоритму перетворення скалярних інструкції в векторні такий [2]:

1) необхідність існування відповідної векторної інструкції в системі команд;
2) складність алгоритму знаходження та перетворення у векторні інструкції;
3) ефективне застосування обмежено великими векторами (від 1024 елементів).

Крім універсального підходу існує спеціалізований, який розрахований на короткі вектори (64 елементи). Цей підхід має спрощений алгоритм перетворення і виконує перетворення циклів. Крім першого обмеження універсального алгоритму він має такі:

1) відсутність залежності між ітераціями циклу;
2) вхідні дані мають бути впорядкованими.

Типи циклів непридатні для векторизації [1]:

1. Цикл з умовним переходом в тілі циклу.
2. Цикл з безумовним переходом в тілі циклу.
3. У тілі зустрічається оператор переривання циклу.
4. У тілі є виклик функції.
5. У тілі зустрічається оператор вводу виводу.
6. Змінна циклу визначається в тілі циклу.
7. Межі циклу визначаються в тілі циклу.

В якості характеристик циклу для визначення можливості векторизації можуть бути при фіксованій архітектурі комп'ютера [1]

1. Кратність кількості векторних регістрів.
2. Мінімальний розмір тіла циклу, який визначається часом виконання векторної інструкції.

Постановка задачі

Розглядаючи сучасні підходи до побудови конфігурованої системи, формулюємо задачу виконання оптимізацій перетворень коду вхідних програм з метою отримання підвищення продуктивності системи. Зважаючи на те, що значний час виконання програми припадає на виконання інструкцій в межах вкладених циклів, перетворення цих інструкцій може мати найбільший вплив на загальний час виконання програми. Використання можливості зміни системи команд конфігурованої системи та використання підходів з перетворення команд внутрішніх циклів у векторні інструкції дозволить отримати підвищення продуктивності виконання внутрішніх циклів. У результаті постає задача створення алгоритму перетворення інструкцій внутрішнього тіла циклу у векторну інструкцію конфігурованої системи.

Загальні підходи та рішення

Здебільшого більше ніж половина процесорного часу затрачається на оброблення циклів. Серед вкладених циклів найбільш вкладений цикл використовується, здебільшого, найінтенсивніше. З огляду на вищенаведене в цій роботі зосереджено увагу на перетворення у векторну інструкцію саме найбільше вкладеного циклу. Окрім впливу на продуктивність виконання прог-

рами, в циклі зазвичай обробляються однорідні дані. Цикл є набір інструкцій, які повторюються, отже, тіло циклу можна перетворити на одну векторну інструкцію.

Створення інструкцій, які відповідають тілу циклу, дасть змогу отримати такі переваги:

1) прискорення виконання – спеціалізована інструкція буде виконуватися швидше за послідовність стандартних інструкцій;

2) скорочення коду програми – економія використання пам'яті, час звернення до пам'яті, локалізація даних;

3) видалення спеціалізованих команд циклу, як команди переходу, обчислення ітераційних змінних.

Серед можливих недоліків можна виділити:

1) неоднорідність архітектури команд процесора – складні інструкції будуть вимагати більшого часу виконання, отже, можливе сповільнення виконання інструкцій, оскільки час такту прирівнюватиметься до часу найповільнішої інструкції. Цей недолік може бути усунений за рахунок використання конвеєризації виконання інструкції;

2) ускладнення алгоритму перетворення, що вимагає збільшення часу перетворення. Цей недолік не є критичним для системи, яка не використовує динамічну компіляцію.

Основними критеріями вибору тіла циклу для перетворення у векторну інструкцію є:

1) відсутність інструкцій вводу/виводу. Оскільки кількість портів вводу/виводу обмежена, тому застосування векторної інструкції є недоцільним;

2) розмір тіла циклу має бути достатнім, щоб використання векторної інструкції було виправдано. Мінімально кількість ефективних інструкцій в тілі циклу має бути >2. Під ефективними інструкціями вважатимемо інструкції, які не є службовими, як інструкції керування, пересилок, звернення до пам'яті, обчислення ітераційних змінних, адрес пам'яті.

Для виконання перетворення тіла циклу необхідно здійснити деякі перетворення самого циклу. Види оптимізацій циклів для проведення векторизації [1]:

1) розбиття циклу (Strip mining);
2) розподіл циклу (Distribution);
3) перестановка циклів (Loop interchange);
4) виведення за межі циклу операції збереження(Hoist and sink);
5) виведення за межі циклу граничних присвоєнь (Peeling);
6) видалення зайвих перевірок (Redundant test elimination);
7) виведення за межі циклу умовного оператора (Test promotion);
8) розгортка циклу (Unroll);
9) оптимізація звернень до пам'яті;
10) динамічний вибір способу виконання циклу.

У цій роботі застосовували 8 тип для перетворення циклу в послідовність інструкцій, 5,6, 9 тип для зменшення тіла циклу. 1,2 тип можна застосовувати для циклів з великою кількістю інструкцій та оптимізації використання апаратних ресурсів.

Алгоритм визначення та утворення векторних інструкцій

У цій роботі розглядаються цикли, які відповідають вимогам та обмеженням, наведеним вище. Процес перетворення складається з двох основних етапів. Перший етап включає виділення тіла циклу, визначення його придатності до перетворення. На другому етапі виконується перетворення циклу у векторну інструкцію, формування внутрішній інструкцій векторної інструкції, виконання оптимізаційних перетворень циклу та внутрішніх інструкцій. Внутрішніми інструкціями будемо вважати інструкції, виділені з тілу циклу, на основі яких будується спеціалізована векторна інструкція. Формування векторної інструкції відбувається конфігуруванням відповідного обчислювального блока конфігурованого ядра [3]. На вхід блока утворення векторної (рис. 1) інструкції надходить код програми мовою проміжного рівня. В якості мови проміжного представлення може бути внутрішня мова компілятора, ассемблерна мова чи будь-яка інша. У нашому випадку можна використати трьохадресний код [4]. У результаті отримується код мовою проміжного представлення з векторними інструкціями та окремий код з внутрішнім представленням кожної векторної інструкції.

Після того, як в коді програми буде визначено наявність циклу, виконується його аналіз на предмет можливості перетворення тіла циклу у векторну інструкцію. Якщо цикл задовольняє умови, виконується його розгортка, але замість копіювання тіла циклу формується множина векторних інструкцій. У результаті буде отримано код програми з векторними інструкціями, але ці інструкції не оптимізовані, тобто мають множину інструкцій, які не використовуються або використовуються неефективно. Отже, останнім кроком алгоритму є оптимізаційні перетворення (рис. 2) та утворення нової послідовності коду.

Рис. 1. Загальна блок-схема перетворення у векторні інструкції

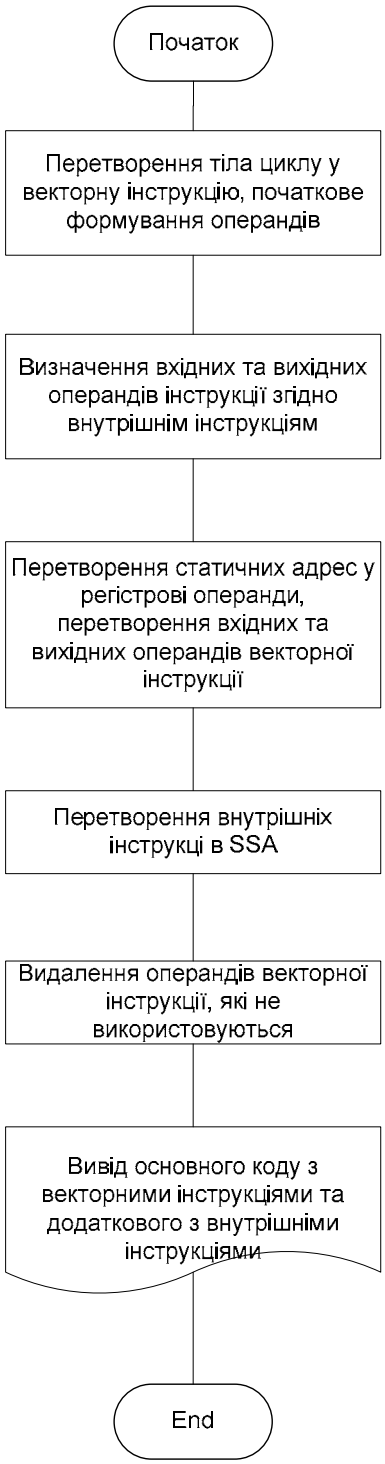

Рис. 2. Структура оптимізаційних перетворень векторної інструкції

Оптимізаційні перетворення можна розділити на внутрішні, коли виконується перетворення внутрішніх інструкцій векторної команди, та зовнішні, коли перетворення зазнає сама векторна інструкція. Серед внутрішніх оптимізаційних перетворень є перетворення в SSA форму, перетворення статичних адрес, видалення внутрішніх інструкції, які не використовуються. До зовнішніх можна зарахувати формування вхідних та вихідних операндів згідно з зовнішнім перетворенням, тобто заміна адрес на фіксовані змінні, видалення операндів, результати яких не використовуються (рис. 2). Зовнішні оптимізації впливають на внутрішні, так видалення зайвих вхідних та вихідних операндів може призвести до видалення відповідних внутрішніх інструкцій. Зовнішня оптимізація відбувається в складі загальної оптимізації програми, а внутрішня – це окремий оптимізаційних прохід.

Перевагою такого алгоритму є:

1) зменшення часу на оптимізацію циклу, тіло оптимізується лише один раз при оптимізації внутрішніх інструкції;

2) алгоритм органічно поєднується з загальною системою оптимізації програми;

3) немає обмеження на типи векторних інструкції, адже вміст циклу визначає векторну інструкцію;

4) може бути застосовано, як для великих векторів даних, так і малих;

5) немає обмеження по залежності між ітераціями, кожна ітерація це окремий виклик однієї векторної інструкції.

Простота виділення векторних інструкцій, векторна інструкція перебуває в межах циклу.

Висновки

У роботі запропоновано загальний підхід до створення векторних інструкцій для конфігурованої системи на базі вкладеного циклу. Визначенні основні обмеження методу з векторизації для фіксованого набору інструкцій та конфігурованого набору інструкцій. Визначені основні переваги векторизації під час використання конфігурованої системи. Наведено алгоритм перетворення частин послідовного коду у векторні інструкції.

1. Крутиков М. П. Оптимизация программ под архитектуру CONVEX C [ел. видання]// http://www.csa.ru/CSA/tutor/opti/index.htmr. 2. Volkonski V.Y., Drozdov A.Y., Rovinsky E.V. "Method of vector operations usage in an optimizing compiler" – Information technologies and computer systems, 3, 2004. (http://www.optimitech.com/005.htm) 3. Melnyk A., Salo A. Automatic generation of ASICs. // NASA/ESA Conference on adaptive hardware and systems. 5–8 August 2007, Edinburg, UK. – P. 311–317. 4. Клименко В. А. Опрацювання статичних масивів в архітектурі на основі пам'яті з впорядкованим доступом [Текст] / В.А. Клименко // Вісник Нац. ун-ту "Львівська політехніка": № 630: Комп'ютерні системи та мережі. – Львів: Вид-во Нац. ун-ту "Львівська політехніка". – 2008. – С. 61–66.