УДК 004.415.2.043

САЛАПАТОВ В.І.

ПІДГОТОВКА АДРЕСИ ОПЕРАНДІВ ПРИ СИНТЕЗІ КОДІВ ПРОГРАМ

У статті розглядається метод підготовки операндів команд під час синтезу машинного коду команд у компіляторах. Пропонується узагальнена модель даних для інформаційних ресурсів. Розроблена принципіально нова схема алгоритму синтезу оптимального машинного коду програм для цільових ЕОМ.

In the article a method of preparation operands at the synthesis of machine code commands in compilers is discussed. A generalization of the data model for information resources is proposed. It's developed a fundamentally new scheme of the synthesis of optimal machine code programs for the target computer.

Вступ

Формування кодів команд у процесі синтезу кодів програм вимагає підготовки адрес операндів. Для цього треба використовувати або вже існуючі данні з інформаційних ресурсів, які вже присутні у програмно доступних регістрах (ПДР), або спеціально формувати відповідні команди пересилок для підготовки необхідних для формування кодів команд адрес. Традиційні методи компіляції передбачають розміщення операндів у певних ПДР, що є однією з причин надмірності програмного коду. Адже для створення оптимальних за певними критеріями програм на рівні машинного коду треба використовувати усі наявні на кожен момент виконання програм данні. Обмеження стосовно використання ПДР при компіляції породжує низку зайвих команд пересилок даних, частка яких є досить значною. Тому дуже важливо використати у повній мірі як вже існуючі інформаційні ресурси (ІР), так і оптимальні за форматом та використанням ПДР команди підготовки адрес операндів для машинних команд. Така машинно-залежна оптимізація коду програм вимагає нового підходу щодо формування машинних кодів. На відміну від звичайної генерації машинного коду за жорсткими правилами маємо синтез кодів з максимальним використанням ІР та системи команд цільової ЕОМ (ЦЕОМ) разом з ПДР цільової ЕОМ. Саме так створюються стислі програми на мовах Асемблер. Але програмування на Асемблері досить трудомістке і вимагає значних зусиль та часу. Компілятори мов програмування полегшують створення програм, але все ще вносять значну надмірність коду програм. Тому проблема підвищення продуктивності комп'ютерних систем за рахунок підвищення якості машинних кодів програм є актуальною і являє собою як теоретичний, так і практичний інтерес.

Огляд особливостей внутрішнього подання програм

Для реалізації машинно-залежної оптимізації кодів програм з максимальним використанням наявних ІР та системи команд ЦЕОМ з одного боку треба мати опис системи команд та особливостей архітектури ЦЕОМ [1, 2], а з другого – зберігати та постійно оновлювати дані про ІР. Процес компіляції програм з мов програмування високого рівня із застосуванням методу машинно-залежної оптимізації передбачає наявність таких даних: початкова програма користувача, опис системи команд ЦЕОМ, опис ІР та вибір форми подання ІР. У процесі синтаксичного розбору програми вона перетворюється у низку внутрішніх уявлень. Кінцеве уявлення повинно складатися з низки операторів, які за своєю семантикою наближені до системи команд ЦЕОМ, але без прив'язки даних до конкретного типу адресації з посиланням тільки на імена змінних та констант. Треба зазначити, що компілятори використовують тільки частину машинних команд ЦЕОМ. Це перш за все арифметико-логічні команди, команди пересилок даних, команди зсуву та команди розгалуження. Решта машинних команд використовуються стандартними та спеціалізованими бібліотечними підпрограмами. Крім того, для внутрішнього подання треба, також, враховувати наявність розширених арифметичних команд. Так, для сучасних комп'ютерів Intel та інших це пов'язано з наявністю співпроцесора, де на рівні мікропрограм реалізовано арифметичні операції над дійсними даними та обчислення деяких функцій. Сигнальні процесори родини ADSP-21xx реалізують однією командою виконання обчислення A+B*C і можуть при цьому суміщати до двох пересилок даних. Це також треба враховувати для кінцевого внутрішнього подання програми. Таким чином, необхідно ви-

значити ті оператори внутрішнього подання, які за своєю семантикою присутні у системі команд ЦЕОМ. Далі, треба обрати форму подання даних про ІР. У подальшому необхідно обрати спосіб формування асемблерних операторів ЦОЕМ, які відображають фактично машинний код. Перетворення програм в їх асемблерний еквівалент використовується майже усіма сучасними компіляторами і є досить зручним як при реалізації, так і при аналізі.

Перетворення внутрішнього уявлення програм в асемблерний еквівалент

Сигнальні процесори родини ADSP-21xx мають нестандартні арифметичні команди, які дозволяють прискорити обчислення, перевірити умову їх виконання та сумістити ще до двох пересилок. Тобто, рівень машинних команд наближається до рівня мікропрограмування. З наявно існуючих груп команд ЦЕОМ треба обрати ті, що будуть відображатись внутрішніми операторами компілятора. З групи арифметико-логічних команд обираються лише ті, які не потребують додаткового обчислення. Наприклад, обчислення функції сінусу у команді співпроцесора для комп'ютерів Intel пов'язано з обмеженим діапазоном аргументів і вимагає перерахунку значення аргументу, тому такі команди використовуються тільки у бібліотечних підпрограмах. Таким чином, арифметикологічні оператори внутрішнього уявлення охоплюють семантики тільки тих машинних команд, які одразу дають остаточний результат ALU = $S(M_{al})$ без попереднього перерахунку. ALU – це семантики арифметико-логічних операто-рів внутрішнього уявлення, S функція взяття семантики, M_{al} – машинні команди, які дають кінцевий результат. Команди пересилок повинні охоплювати пересилки даних та адрес у форматах регістр-регістр, регістр-пам'ять, пам'ять-регістр, регістр – безпосередній операнд та пам'ять-пам'ять. Тобто MOV = $S(M_{rr}, M_{rm}, M_{mr}, M_{ri}, M_{mm})$. MOV – це семантики операторів пересилок внутрішнього уявлення, M_{rr}, M_{rm}, M_{mr}, M_{ri}, M_{mm} – команди пересилок усіх перелічених вище форматів. Ці команди використовуються при компіляції як допоміжні для підготовки операндів для інших команд. Команди зсуву часто використовуються у виразах, тому доцільно включити внутрішні оператори з семантикою зсуву у перелік операторів внутрішнього уявлення. До них відносяться арифметичні, логічні та циклічні оператори зсуву праворуч та ліворуч. Отже, SHIFT = $S(S_{ar}, S_{al}, S_{lr}, S_{ll}, S_{rr}, S_{rl})$, де S_{ar}, S_{al}, S_{lr}, S_{ll}, S_{rr}, S_{rl} – машинні команди правого та лівого зсувів арифметичного, логічного та циклічного відповідно, а SHIFT – семантики операторів зсуву внутрішнього уявлення. І, насамкінець, команди розгалуження, до яких відносяться команди безумовного та умовного переходів та зациклювання за шістьома умовами ($<$, $\leq$, $>$, $\geq$, $=$, $\neq$) з урахуванням знаку та без нього. Сюди ж треба віднести команди звернення до підпрограм та повернення з них, або CHANGE = S(Jmp, Jxx, Bxx, LOOP, Lxx, CALL, RET). CHANGE – це множина семантик операторів розгалуження внутрішнього уявлення. Jmp, Jxx, Bxx, Loop, Lxx, CALL, RET – це відповідні машинні команди ЦЕОМ розгалуження.

Стосовно подання опису машинних команд, то раніше вже було обрано спеціальну структуру бази даних (БД), за допомогою якої повністю описуються семантики кожної машинної команди [1, 4].

Залишилось обрати форму подання ІР. Для простоти звернення та наступної обробки ІР треба подати у вигляді реляційної БД. Ключовими елементами головного відношення мають бути змінні або елементи масивів певного типу. Решта атрибутів кожного запису таблиці визначає тип даних, його довжину та спосіб його розміщення через ПДР або стек. У відповідності до цього структура такої реляційної таблиці TABIP з ІР має наступний вигляд.

Ім'я змінної	Тип змінної	Довжина	Розміщення	Ідентифікатор
NAME	TYPE	LEN	ADDR	ID

Рис. 1. Структура відношення опису ІР

Якщо змінна являє собою елемент масиву, то для вказівки на сукупність індексів масиву до основного відношення необхідно додати ще відношення, яке вказує на елемент масиву за сукупністю його індексів. Структура такого відношення INDEX подана на рисунку 2 нижче. Атрибути ID та NDX є ключовими і пов'язують елемент масиву за сукупністю значень усіх його індексів. Розмірність масиву визначає максимальну кількість записів у цій таблиці. Кількість записів для одного елементу відповідає розмірності

масиву. Зв'язок з основним відношенням – один до багатьох.

Ідентифікатор	Номер індексу	Значення індексу	MAX індексу
ID	NDX	NUM	MAXNDX

Рис. 2. Структура відношення для даних типу масив

Для вказівки способу розміщення операнда як IP необхідно ще одне відношення, яке б вказувало на спосіб адресації до наявного IP. Атрибути ID та ADDR є ключовими і пов'язують елемент масиву за способом його розміщення (ПДР, пам'ять, вид адресації у пам'ять). Атрибут TREG вказує на тип ПДР (базовий, індексний, тощо). Ім'я ПДР визначає конкретний ПДР. Структура відношення LOCATE, яке визначає розміщення IP показана на Рис. 3. Атрибути ID та ADDR є ключовими. Зв'язок з основною таблицею – один до багатьох.

Ідентифікатор	Розміщення	Тип ПДР	Ім'я ПДР
ID	ADDR	TREG	NAMREG

Рис. 3. Структура відношення розміщення даних

Якщо потрібний IP для формування асемблерного оператора є у наявності, то через відношення TABIP він буде знайдений. У разі масиву адреса операнда через відношення INDEX перераховується у відповідності із значеннями індексів та формується додатковими командами.

У разі відсутності IP необхідно сінтезувати команди пересилки для доступу до потрібного операнда. Блок-схема алгоритму синтезу асемблерних операторів подана на Рис. 4.

Алгоритм являє собою цикл аналізу операторів внутрішнього подання та синтезу асемблерних операторів програми користувача. Цикл закінчується по досягненню останнього оператора. У процесі аналізу знаходиться машинна команда з відповідною семантикою та готуються операнди. Якщо операнд присутній у відношенні TABIP, то відбувається синтез оператора, інакше формуються допоміжні асемблерні оператори по розміщенню потрібних операндів або їх адрес у ПДР. Далі у циклі відбувається пошук та формування наступних операндів, що вказані в операторі внутрішнього подання аж до остаточного формування оператора Асемблера ЦЕОМ. Якщо машинна команда має кілька форматів, то обирається найкращий за визначеним критерієм оптимальності [3, 4, 5]. У відношенні TABIP код розміщення може вказувати на стек з вказівкою на відповідний номер. При цьому при запису чергового даного у стек усі номери елементів стеку збільшуються на 1, а при читанні зі стеку – номери зменшуються на 1. З урахуванням довжини операнда у стеку обчислюється його фактична адреса. Таким чином, якщо після машинно-незалежної оптимізації внутрішнього подання вважається, що програма оптимальна, то її машинний код також буде максимально наближений до оптимального. Чому ж тільки наближений? Тому що тут не враховуються різні додаткові програмістські хитрощі, які не враховані у цьому загальному алгоритмі. Це пов'язано з деякими особливостями архітектури та системи команд конкретної ЦЕОМ. Компілятор мови С для сигнальних процесорів родини ADSP-21xx, який породжує машинні коди за обраною схемою, враховує деякі специфічні команди, що дозволяють об'єднати кілька дій в одній команді. Так, наприклад враховується наявність машинних команд типу A+B*C з одночасними пересилками даних. Фірмовий компілятор породжує кілька команд і не враховує особливостей архітектури та системи команд сигнального процесора.

Треба зазначити, що у блоці синтезу команд пересилок постійно виконуються дії по оновленню БД IP.

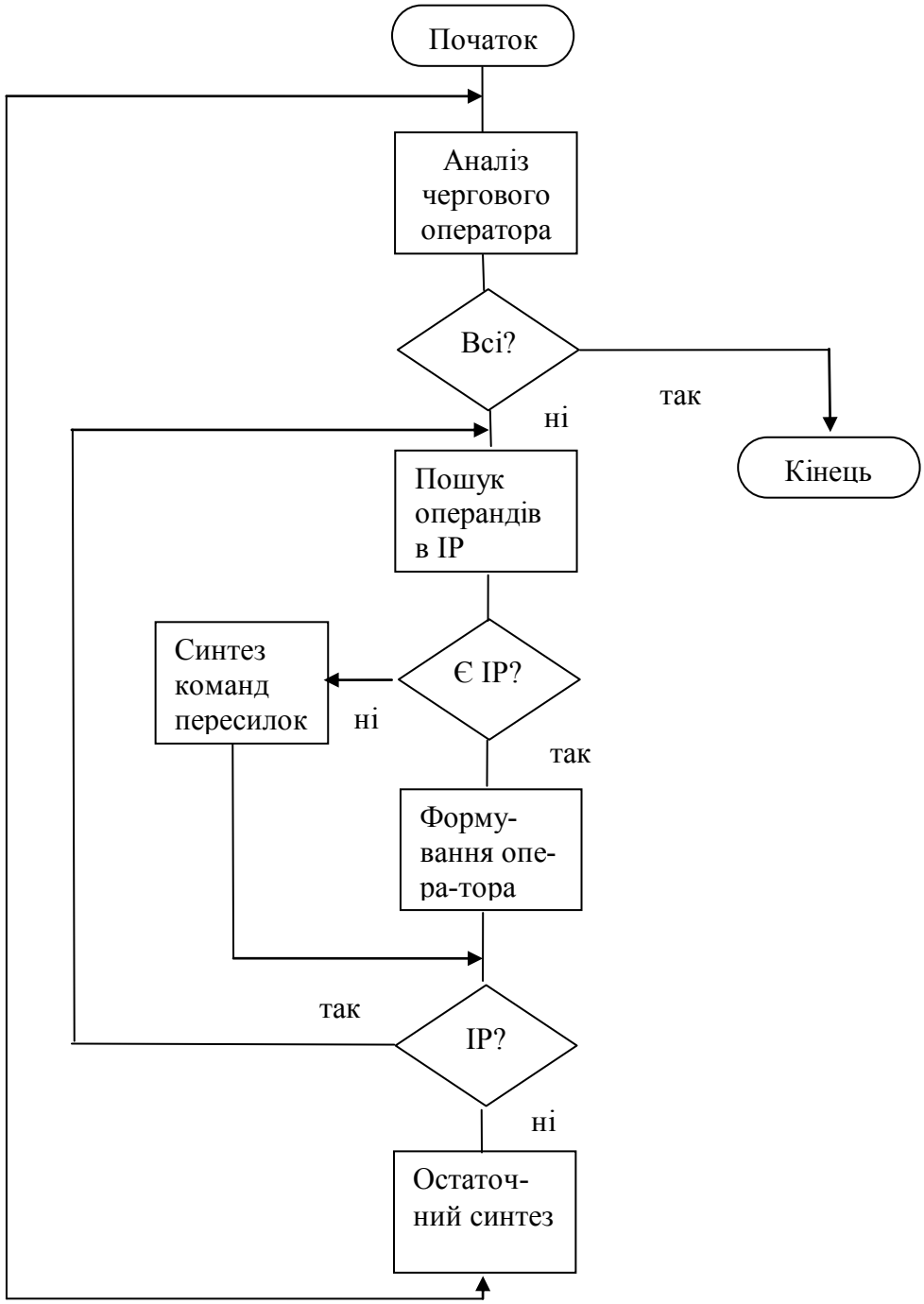

Рис. 4. Блок-схема алгоритму синтезу кодів команд

Висновки

Запропонована узагальнена модель даних ІР, а також принципово нова схема алгоритму синтезу машинних кодів програм. Це дозволяє здійснювати у компіляторах машинно-залежну оптимі-зацію кодів програм за обраним критерієм під час синтезу машинних кодів. Визначені групи машинних команд, які використовуються компіляторами та обрані оператори внутрішнього уявлення компіляторів.

Для подання даних про ІР обрано реляційну модель, яка дозволяє формалізувати операції над даними за допомогою операцій реляційної алгебри і формувати оптимальний асемблерний еквівалент програми користувача. Така схема синтезу машинного коду програм може бути застосована для різних компіляторів різних ЦЕОМ. При цьому за рахунок ускладнення схеми компіляції на виході маємо оптимальний код програми користувача у вигляді асемблерного тексту. Далі для отримання об'єктного файлу текст програми обробляється транслятором з Асемблера ЦЕОМ. Стосовно особливостей архітектури та системи команд конкретної ЦЕОМ, то вони мають бути враховані у схемі

синтезу кодів команд окремо. Запропонована схема синтезу коду програм може бути впроваджена в компіляторах взагалі для ЦЕОМ з довільною архітектурою.

Список літератури

1. Салапатов В. І. Структура данных для описания семантики и синтаксиса команд целевой ЭВМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 41. Київ. 2004 с. 191-198.
2. Машинно-зависимая оптимизация кодов программ при их синтезе. ж. Электронное моделирование. № 2003. с. 33-44.
3. Салапатов В. І. Використання реляційної моделі даних для внутрішнього уявлення програми. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 42. Київ. 2004 с. 191-198.
4. Салапатов В. І. Подання даних для синтезу коду у нормалізованому вигляді. Вісник Черкаського інженерно-технологічного інституту, 2001, №3, с.96-99.
5. Салапатов В. І. Особливості підвищення якості програм у сигнальних процесорів. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 52. Київ. 2010. с. 75-79.