

РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ ПРО ВНУТРІШНЄ УЯВЛЕННЯ ПРОГРАМ

В цій статті запропоновано реляційну структуру даних для внутрішнього подання програм. Це дає можливість формалізувати процес машинно-незалежної оптимізації програм із застосовуванням операцій реляційної алгебри та передати оптимізоване внутрішнє подання програми на останню фазу компіляції – синтез машинних кодів.

There are proposed the relation data structure for internal program imaging in this article. That give possibility formalize the machine independed optimization program process due to relation algebra operations and pass the internal optimized program imaging to the last phase of compiler work – machine code synthesis.

Класичний трьохадресовий код, що використовується для подання внутрішнього уявлення програм [1, 2], зручний лише для виразів, а для таких конструкцій, як присвоєння, цикли та розгалуження через свою жорстку структуру він є надмірним і тому не зовсім зручним. Реляційна модель подання даних дозволяє визначити для кожної конструкції мови семантику складових операторів, операнд *sd* разом з їхніми типами, а також місце розташування результату без жорсткої прив'язки до певних шаблонів. Крім того, стандартні реляційні операції над відношеннями дозволяють достатньо просто виконати машинно-незалежну оптимізацію програм з використанням операції пошуку за зразком [3]. Таким чином, внутрішнє подання програм доцільно представити у вигляді нормалізованої реляційної БД, яка максимально компактно та стисло подає інформацію без втрат і усуває суперечність даних. Тому буде цілком доречно застосувати таку модель даних і для внутрішнього подання програм та в подальшому використовувати реляційні операції на фазі машинно-незалежної оптимізації [3]. ER-модель внутрішнього уявлення програми має враховувати крім семантик операторів та операндів з їхніми атрибутами ще й порядковий номер оператора. З урахуванням цього така модель виглядає так, як наведено на малюнку 1. Сутність TOPUP відображає оператор внутрішнього подання. Через параметр SEM встановлюються зв'язок з сутністю OPERAND, яка визначає для оператора із заданою семантикою припустимий список операндів з атрибутами: TOP – типом операнда (приймальник, джерело тощо) та LTD – обмеження значення операнду. Для верхівки TOPUP визначається конкретний операнд (сутність DATA) через зв'язок PR та VN між сутностями OPERAND та DEFINE через сутність DATA. Безпосередній зв'язок конкретного опе-

ранду DATA з верхівкою TOPUP здійснюється через номер оператора NUM і далі через ім'я змінної або константи VN – із змінними та константами (сутність DEFINE).

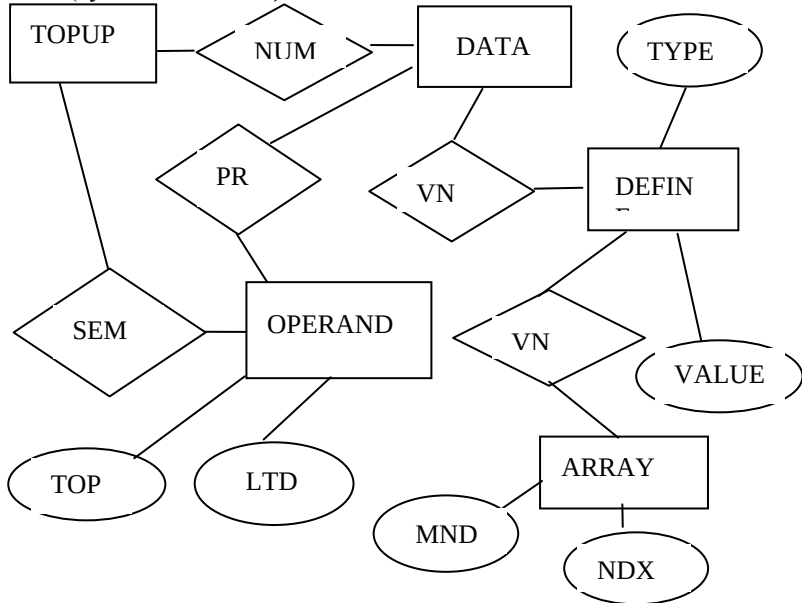


Рис.1. ER-модель даних про внутрішнє уявлення програми

Властивості TYPE та VALUE визначають тип операнду та його поточне значення. Сутність ARRAY через зв'язок VN пов'язана із сутністю DEFINE – ім'ям змінної та використовується, якщо операнд являє собою елемент масиву. При цьому атрибути MND та NDX вказують на номер індексу масиву та максимальне значення індексу відповідно.

Ідеальним варіантом щодо семантик операторів внутрішнього подання програм є наявність операторів машинних команд з такими самими семантиками в системі команд EOM. А такі складні конструкції, як цикли та розгалуження можуть подаватися у вигляді низки операторів з елементарними семантиками [4]. Таким чином, передбачається подальше перетворення синтаксичного дерева в процесі його обходу, тобто аналізу, у низку операторів з певними елементарними семантиками. До таких семантик можна віднести арифметичні та логічні операції над цілими, дійсними і символічними даними, операції пересилки, операції зсуву, операції умовних та безумовних переходів і деякі специфічні операції (перекодування, пошук розподільника і т.і.). Як вже було зазначено [4, 5], в разі відсутності команди з певною семантикою в системі команд EOM для неї може бути передбачено спеціальне макровизначення. Для подання внутрішнього уявлення програми необхідно пов'язати номер оператора із

заданою семантикою з оператором, що описується сутністю **TOPUP**. Тому у базовому відношенні основним ключовим атрибутом, що являє собою сутність оператора, обрано номер оператора NUM. Код SEM семантики вказує на тип операції внутрішнього оператора і в подальшому використовується для перетворення синтаксичного дерева у внутрішній код. Тому для внутрішнього подання програм таких даних нижче наведено відношення **TOPUP** із структурою, яка відображена на рисунку 2.

TOPUP	
NUM *	Номер верхівки
SEM *	Код семантики

Рис. 2. Структура відношення опису верхівок синтаксичного дерева

З кожною такою верхівкою дерева синтаксичного розбору мають бути пов'язані підлеглі верхівки – операнди. Ці підлеглі верхівки можуть бути кінцевими (листами) і тоді вони вказують безпосередньо на операнд (змінну або константу), або вказувати на іншу операцію і тоді вони вказують на верхівку типу TOPUP. Семантика кінцевої верхівки передбачає наявність конкретних операндів і тому потребує додаткового відношення для них. Для подальшого синтаксичного аналізу коди семантик кінцевих верхівок мають відрізнятися від кодів семантик операторів. Для таких кодів, що являють собою операнди, необхідно спеціальне відношення для опису даних. Це відношення повинно відображати константи та змінні, які були визначені у програмі користувача. Останні визначаються у програмі за допомогою декларативних операторів визначення змінних, констант і т.і. Для даних типу **масив** мають бути враховані індекси та їхні максимальні значення. Для даних типу **запис** необхідно ще вказати тип кожного елементу запису або його довжину та вказівник на елемент в середині запису. Остання вимога легко реалізується при використанні составного імені. Таким чином, кожен елемент запису визначається як звичайна змінна певного типу з певною довжиною. З оглядом усього сказаного нижче, на рисунку 3. представлена структура відношення **DEFINE** для визначення даних програм разом з їх типами та значеннями.

DEFINE	
VN *	Ім'я змінної
TYPE	Тип змінної
VALU	Значення
E	

Рис. 3. Струк-

тура відношення опису даних програми користувача

тура

Ім'я змінної VN разом з її типом TYPE та початковим або поточним значенням VALUE разом з можливими індексами повністю описують

прості дані програми. Ключовий параметр VN дозволяє зв'язати ім'я масиву з його параметрами (кількістю індексів та їхніми максимальними значеннями). При визначенні змінних, масивів та констант у програмі створюється відповідний запис у відношенні **DEFINE**. Під час виконання програми або під час її часткового моделювання при компіляції значення кожної змінної може змінюватися і ці зміни мають враховуватися у відношенні **DEFINE**. В разі визначення даних у вигляді масиву необхідно задати його розмірність та максимальні значення індексів. Значення конкретного елементу масиву має визначатися параметром VN через ім'я разом з усіма індексами. Для контролю значень індексів та їхньої кількості у процесі компіляції необхідно додаткове відношення. Нижче на рис. 4 наводиться структура відношення **ARRAY**, яка дозволяє виконувати контроль за зміною індексів масивів.

ARRAY	
VN *	Ім'я змінної
NDX	Номер індексу (для масиву)
MND	Максимальне значення індексу

Рис. 4. Структура відношення опису додаткових даних для масиву

Ключовий атрибут VN імені масиву пов'язує конкретний масив за його сутністю з номером індексу NDX, починаючи із старшого індексу, та максимальним значенням цього індексу MND. В разі опису структур або масивів структур, що визначається атрибутом TYPE у відношенні **DEFINE**, кожен елемент таких структур визначається як змінна із составним ім'ям, типом та можливим початковим значенням. Фактичне значення кількості записів у масиві структур має відповідати параметру MND у відношенні **ARRAY**. При цьому значення атрибуту NDX має бути рівним 1, тобто вказувати на одновимірний масив.

Для створення компілятора мови програмування необхідно обрати стандартний за семантиками набір операторів внутрішнього подання програм [6]. Цей набір складається із стандартних арифметико-логічних операторів для цілих і дійсних даних, операторів зсуву, пересилок, деяких спеціальних операторів, операторів розгалуження та операторів введення-виведення. Кожен з цих операторів так чи інакше пов'язаний з певними операндами. Для того, щоб задати синтаксис операторів мови програмування необхідно визначити правила граматики такої мови. З точки зору синтаксичної конструкції кожного з операторів мови програмування правила граматики визначають коректність операндів у семантичних верхівках при побудові синтаксичного дерева. Оскільки за основу подання даних в роботі було обрано реляційну модель, то цілком логічним є використання цієї моделі і для визначення синтаксичних правил мови програму-

вання. Так, якщо в деякій мові програмування маємо якусь обмежену множину операторів з їх семантичними верхівками S_i , то необхідно визначити операнди для цих операторів OP_j^i як підлеглі до них верхівки разом з типами, порядком розташування та можливими обмежувальними значеннями. З метою наближення семантик внутрішніх операторів мови до семантик машинних команд ЕОМ необхідно із загальної множини стандартних семантик S внутрішніх операторів мови обрати семантики тих команд, які присутні в системі команд ЕОМ. Таке наближення суттєво спрощує роботу компілятора при породженні машинних кодів програм [5]. Для спеціальних мовних конструкцій, що визначають тип оператора мови (умовний оператор, оператори циклу різного виду, тощо) на фазі синтезу кодів програм цілком припустимим є використання макросів. Відповідні макровизначення, які необхідно заздалегідь підготувати, мають відповідати семантиці такого оператора і повинні бути внесені до списку семантик команд ЕОМ.

Як вже було зазначено, трьохадресовий код для внутрішнього подання програм, у який перетворюється синтаксичне дерево програм і який широко застосовується в сучасних компіляторах, має стандартну жорстку структуру і для багатьох операторів є надлишковим і тому не зовсім зручним для подальшої машинно-незалежної оптимізації програм [7]. До того ж, кожен оператор внутрішнього подання програм має свій власний формат і може бути представлений як частинка графу, у верхівці якого знаходиться оператор з певною семантикою, а гілками його є операнди. На рисунку 5 показано структуру відношення **OPERAND**, що фактично описує семантичні правила щодо операндів в операторах мови програмування.

OPERAND	
SEM	Код семантики оператора
*	
PR	Порядковий номер операнду
*	
TOP	Тип операнду
LTD	Обмеження значення операнду

Рис. 5. Структура відношення опису операнду для оператора внутрішнього подання програм

За даною логікою кожен оператор із певною семантикою може мати стільки записів, скільки в нього є операндів. Це робить таке уявлення дуже гнучким та зручним для внутрішнього подання програми за допомогою БД. Набір базових семантик внутрішнього подання операторів разом з описом їхніх операндів можна вважати постійним або умовно постійним. Зміни у наборі базових семантик можуть вноситись по мірі подальшого розширення та удосконалення систем команд ЕОМ.

Але структура БД, що формується при синтаксичному аналізі для внутрішнього подання програм, має враховувати прив'язку операторів до операндів та їхній опис щодо конкретного елементу внутрішнього подання програми.

Для зв'язку внутрішніх операторів з конкретними даними потрібно відношення, що відповідає сутності DATA в ER-моделі даних про внутрішнє уявлення програми. На малюнку 6 наведена структура даних такого відношення DATA.

DATA	
NUM *	Номер оператора
PR *	Код призначення операнду
VN *	Ім'я змінної

Рис. 6. Структура відношення зв'язку оператора із змінними

Як бачимо, наведене відношення повністю складається з ключових атрибутів і має використовуватися з одного боку для зв'язку операторів з конкретними операндами, а з іншого боку через зв'язок PR – з відповідним операндом відношення OPERAND, що дозволяє контролювати наявність операнда згідно з його призначенням.

Повна структура БД для опису внутрішнього уявлення програми з наведеними вище відношеннями має вигляд, що представлений на рис. 7.

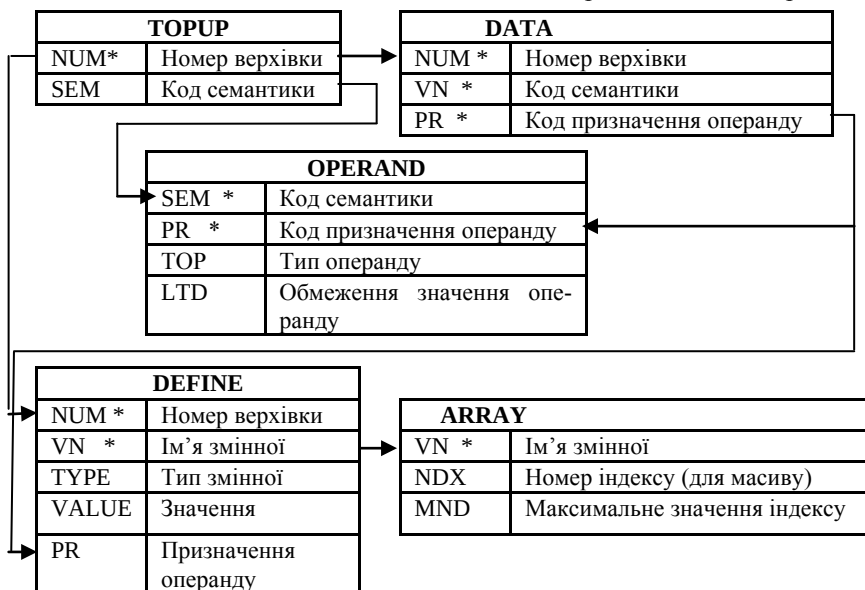


Рис. 7. Структура БД для опису внутрішнього уявлення програми

Даний малюнок відображає структуру БД для внутрішнього уявлення операторів мови. Верхівка дерева, що описується відношенням TOPUP, за своїм номером NUM вказує на конкретну змінну з іменем VN, типом змінної TYPE, її призначенням PR та поточним значенням VALUE. Для комутативних операторів порядковий номер операнду не має значення і визначається кодом семантики SEM. Відношення **OPERAND** фактично частково визначає семантику оператора мови, тому невідповідність типів операндів вказує на семантичну помилку. Якщо операнд є елементом масиву, то відношення **ARRAY** визначає його розмірність та обмеження на індекси.

Спеціальні мовні конструкції за допомогою макросів можуть перетворюватися у оператори виразу, але їхній синтаксис має описуватись також відношенням **OPERAND**, яке визначає типи операндів.

Запропонована структура даних для внутрішнього подання програм користувача дозволяє формалізувати процес їхньої машинно-незалежної оптимізації їа допомогою реляційних операторів [3]. Така структура використовується в компіляторі мови С мікро-контролера AVR, який розроблений на кафедрі кібернетики Черкаського національного університету імені Богдана Хмельницького.

Список посилань

1. А.Ахо, Р.Сети, Дж.Ульман. Компиляторы: принципы, технологии и инструменты. М.: Вильямс, 2001. – 768с.
2. А.В.Гордеев, А.Ю. Молчанов. Системное программное обеспечение. СПб.: Питер. 2001. – 736 с.
3. Салапатов В.І. Еквівалентні перетворення внутрішнього уявлення програм. Вісник ЧДТУ. № 45. Черкаси. 2004. с. 128-132
4. Салапатов В.І. Особливості формування інформаційних ресурсів для різних блоків програм. Вісник Херсонського державного технічного університету, Математичне моделювання та інформаційні технології. 2006. Херсон. с. 141-144.
5. Салапатов В.И., Кузнецов А.В. Структурированный макроассемблер виртуальной ЭВМ. Вестн. Киев. политехн. ин-та. Автоматика и электроприборостроение. К., Вып 22. 1986. с. 28-31
6. Салапатов В.І. Адаптація процесу синтезу кодів програм під систему команд цільової ЕОМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 43. Київ. 2005 с. 75-79
7. Салапатов В.И. Особливості внутрішнього уявлення програм для подальшого синтезу його коду. Вісник ЧІТІ. № 3. 0.30 Черкаси. 2002. с. 78-84.