ОПТИМИЗАЦИЯ КОДА ДЛЯ МИКРОКОНТРОЛЛЕРОВ С ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРОЙ

А.И. Толкачёв

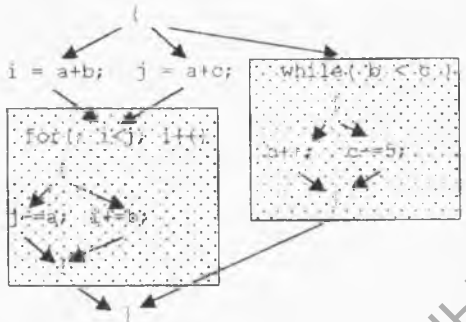
В последнее время наблюдается повышение интереса разработчиков к параллельным архитектурам. Предлагаемый метод анализа алгоритма позволяет в заданной программе на каком-либо языке высокого уровня выделить конструкции, выполнение которых может происходить параллельно.

Рассмотрим работу алгоритма на примере тела функции в языке С. Функция представляется как последовательность операторов. Для простоты предположим, что оператор безусловного перехода *goto* отсутствует. Среди операторов могут быть операторы *for*, *while*, *if*, *switch*, каждый из которых рассматриваются как один оператор с одной точкой входа и одной – выхода. Тела таких операторов так же представляются в виде последовательностей операторов. Очевидно, что при таком представлении каждый блок программы оказывается последовательностью операторов, не содержащей операторов перехода.

Решается задача распараллеливания операторов в пределах одного блока. Для этого для каждого оператора находится множество операторов, результаты работы которых являются его операндами. Оператор может выполняться сразу после того, как отработали все операторы из этого множества.

Рассмотрим пример:

```
{
 i = a+b;
 j = a+c;
 for(; i < j; i++)
 {
  j-=a;
  i+=b;
 }
 while( b < c )
 {
  b++;
  c-=5;
 }
}
```



На правом рисунке изображена схема выполнения программы. Операторы, в которые входит несколько стрелок, выполняются только после того, как все операторы, из которых эти стрелки выходят, закончили работать. Заштрихованными прямоугольниками выделены операторы, которые сами содержат блоки операторов.

В случае присутствия операторов *goto*, последовательность операторов разбивается на множество последовательностей, заключённых между операторами перехода и метками.

Для увеличения качества кода возможно использование алгоритмов, находящих циклы и блоки, не присутствующие явно в программе, а организованные, например, с помощью операторов условного перехода.

Литература:

1. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции: Пер с.англ. // под ред. В. М. Курочкина. - М.: Мир, 1978, Т.1,2.
2. Ф.Льюис, Д.Розенкранц, Р.Стирнз Теоретические основы проектирования компиляторов. М.: Мир, 1976.
3. Terence J. Parr Obtaining Practical Variants of LL(k) and LR(k) for k>1 by Splitting the Atomic k-Tuple // PhD thesis, Purdue University, West Lafayette, Indiana, August 1993.
4. Jacob Navia Lcc-Win32: A compiler system for windows // Technical Reference, based upon the lcc compiler written by C.W. Fraser and Dave Hanson.
5. Толкачёв А.И. Универсальный синтаксический анализатор // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. Материалы IV Республиканской научной конференции студентов и аспирантов 19-22 марта 2001г., С. 194-196.