Савенков К.О., Ющенко Н.В.

МЕТОДИКА ОПИСАНИЯ ПОВЕДЕНИЯ ПРОЦЕССОРА ДЛЯ ОЦЕНКИ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРОГРАММЫ [1]

Московский Государственный Университет,
факультет Вычислительной Математики и Кибернетики, Москва,
savenkov@cs.msu.su, youwhile@cs.msu.su

Обзор

В этой статье мы представляем удобный и достаточно простой подход к описанию поведения процессора для оценки времени выполнения ассемблерной программы. Этот подход удобен для описания конвейерных архитектур и позволяет достичь потактовой точности оценки. Его сложность полиномиальна относительно числа ресурсов процессора и не зависит по памяти от числа инструкций в программе. Подход был успешно применён к процессору NM6403 "Neuromatrix" и по быстродействию значительно превосходит традиционную программную эмуляцию этого процессора.

1. Введение

Задача оценка времени выполнения программы возникает тогда, когда у нас есть программа на языке высокого уровня, предназначенная для выполнения на некоторой целевой машине, и нет доступа к этой целевой машине. В этом случае мы должны оценить время выполнения программы, используя другую (инструментальную) машину.

Пусть у нас есть спецификация архитектуры целевой машины, позволяющая однозначно понять процесс выполнения на ней программы. Будем считать, что архитектура — конвейерная, вследствие чего время выполнения одной инструкции нельзя оценить константой, поскольку инструкции в конвейере влияют на выполнение друг друга.

Для решения этой проблемы мы используем статико-динамический подход [1], общая схема которого представлена на Рис. 1.

На статическом этапе происходит кросс-компиляция программы в язык ассемблера целевой машины, затем в ней выделяются линейные участки, и строятся статические оценки времени их выполнения. В исходную программу (на языке высокого уровня) вносится информация, позволяющая определить трассу (путь) ее выполнения, а затем эта программа компилируется для выполнения на инструментальной машине.

[1] Работа выполнена при частичной поддержке Российского фонда фундаментальных исследований, проект № 01-01-00263.

На динамическом этапе исполняемый файл модифицированной программы запускается на инструментальной машине с конкретным входными данными, и получается трасса выполнения программы. Затем статические оценки для линейных участков объединяются в соответствии с полученной трассой, и, тем самым, вычисляется общее время выполнения программы.

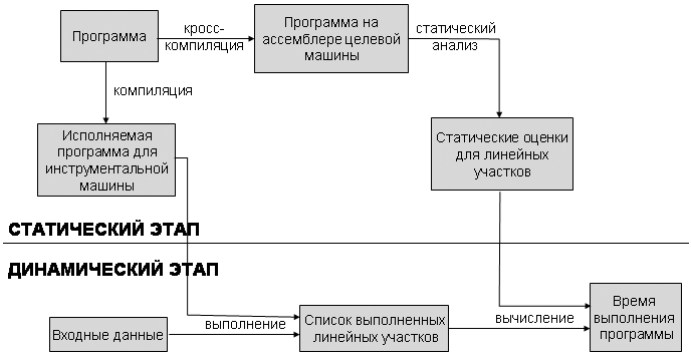

Рис. 1. Общая схема статико-динамической оценки.

Данная работа посвящена получению статических оценок для линейных участков, и последующему динамическому объединению этих оценок в соответствии с известной трассой выполнения программы.

Учитывая то, что линейный участок может состоять из одной инструкции, мы разобьем статический анализ на два этапа: статическую оценку отдельных инструкций и объединение этих оценок в оценку для линейного участка в соответствии с его структурой.

Стоит отметить, что этап статического анализа на практике всегда будет содержать два этих подэтапа, поскольку в программах, как правило, есть последовательности линейных участков, порядок выполнения которых можно определить в статике, в силу чего оценки для них можно объединить также статически.

Наша цель — представить способ, позволяющий производить эту оценку с потактовой точностью.

2. Целевая архитектура

Мы рассматриваем процессоры с достаточно простой схемой организации конвейера.

Инструкции извлекаются процессором из памяти в буфер команд и ставятся в очередь на выполнение. На вершине буфера инструкция декодируется и проверяется наличие всех необходимых для ее выполнения ресурсов. Инструкции проходят через конвейер в порядке

извлечения их из памяти. Это означает, что если один из ресурсов, необходимых для выполнения некоторой инструкции на определенном шаге конвейера, оказался блокированным, то инструкция застрянет на предыдущем шаге, ожидая его освобождения, блокируя тем самым весь конвейер. На одной ступени конвейера может находиться лишь одна инструкция.

Временем выполнения всех инструкций, за исключением последней, считается интервал между запросом на извлечение инструкции и появлением в буфере пустой строки (т.е. временем, когда можно сделать запрос на извлечение следующей инструкции). Для последней инструкции левая граница интервала равна времени завершения её обработки на последней ступени конвейера.

3. Виртуальная модель процессора

Очевидно, что для достижения потактовой точности константных статических оценок времён выполнения отдельных инструкций будет недостаточно, поскольку не будут учтены возникающие в динамике конфликты инструкций по ресурсам. Мы будем использовать оценки, параметризованные временами освобождения ресурсов процессора. Другие особенности процессора, влияющие на выполнение инструкции (например, буфер инструкций, занятость ступени конвейера), мы также опишем в терминах ресурсов - "виртуальных".

Таким образом, мы рассматриваем нечто вроде виртуальной модели процессора, которая управляет выполнением инструкций только при помощи разделяемых или блокируемых ресурсов. Состоянием модели процессора будет набор времен освобождения всех ее ресурсов. Статическая оценка для инструкции (или всей программы) будет абстрактной моделью инструкции (программы), изменяющей состояние модели процессора. В данном контексте, статический этап оценки можно рассматривать как "компиляцию" модели программы, а динамический — как ее абстрактную интерпретацию на модели процессора.

Любой инструкции I ставится в соответствие два списка множеств. Первый содержит множества $use_i(I)$ ресурсов, используемых инструкцией I, для всех i=1…N ступеней конвейера. Второй список содержит множества $block_i(I)$ ресурсов, блокируемых инструкцией I, для всех i=1…N ступеней конвейера.

Для имитации нескольких типов захвата одного физического ресурса используются виртуальные ресурсы. При этом физическому ресурсу ставится в соответствие несколько виртуальных ресурсов, захват одного из которых блокирует остальные.

4. Формальное описание выполнения инструкции

Выполнение инструкции на конвейерном процессоре можно описать как систему правил логического вывода. Некоторые из них приведены на Рис. 2. Например, правило (2) означает, что инструкция прибудет на 1-ю ступень конвейера в момент времени *t*, если 1) все необходимые ресурсы свободны, 2) 1 такт назад эта инструкция поступила на вершину буфера, и 3) предыдущая инструкция уже прибыла на следующую ступень конвейера.

Мы можем записать эту систему в виде системы уравнений, преобразуя предикат $P_k\big|_{t_k} = P_1\big|_{t_1} \wedge ... \wedge P_M\big|_{t_M}$ в уравнение $t_k = \max(t_1,...,t_M)$. С учетом того, что в модели процессора все подобные предикаты соответствуют некоторым виртуальным или физическим ресурсам, мы тем самым выразим время освобождения некоторого ресурса (в левой части уравнения) через времена освобождения других ресурсов.

$$\frac{\bigwedge_{r \in use_N(I)} free(r)\big|_t \wedge stage_{N-1}(I)\big|_{t-1} \wedge stage_{N+1}(pred(I))\big|_p}{stage_N(I)\big|_t}, p \le t$$

Переход на N-ю ступень конвейера

$$\frac{\bigwedge_{r \in use_1(I)} free(r)\big|_t \wedge topbuffer(I)\big|_{t-1} \wedge stage_2(pred(I))\big|_p}{stage_1(I)\big|_t}, p \le t$$

Переход на 1-ю ступень конвейера

$$\frac{stage_1(pred(I))\big|_p \wedge fetched(I)\big|_{t-C_I}}{topbuffer(I)\big|_t}, p \le t$$

Инструкция — на вершине буфера

$$\frac{emptyline()\big|_{t-C_M} \wedge free(bus)\big|_{t-C_M}}{fetched(I)\big|_t}$$

Инструкция выбрана в буфер

Рис. 2. Предикативное описание исполнения инструкции.

Следующий шаг — замена в левой части уравнения времени захвата ресурса на время его освобождение. Мы можем сделать это путем добавления специальной константы к правой части уравнения для конкретного ресурса, поскольку величину данной константы всегда можно определить (на основе известного времени прохождения одной ступени конвейера — 1 такт, и известных задержках на обмен с памятью). После выполнения очевидных подстановок мы получим статическую оценку для инструкции, параметризованную состоянием модели процессора.

5. Матрица оценки

Итак, статическая оценка для инструкции представляется в виде набора формул (по одной на каждый ресурс модели процессора), каждая формула выражает время освобождения одного ресурса через времена

освобождения других ресурсов. Представим подобную формулу в виде вектора, размер которого равен общему числу потенциально блокируемых ресурсов модели процессора. Элементами вектора будут натуральные числа C_r или значения *Null* (которое означает, что освобождение ресурса, соответствующего формуле, не зависит от освобождения ресурса, соответствующего позиции, в которой находится *Null*). Таким образом, формула говорит о том, что время освобождения некоторого ресурса равно максимуму из сумм t_i+C_i для всех ресурсов, от которых зависит выполнение фрагмента программы.

Тогда статическая оценка для инструкции (или абстрактной модели инструкции), может быть представлена в виде матрицы (Рис. 3). Каждый ряд матрицы служит формулой, оценивающей время освобождения конкретного ресурса. Матрица — квадратная, ее размер равен общему числу ресурсов модели процессора.

$$E(I) = \begin{pmatrix} R_1 & \rightarrow & C_{11} & C_{12} & \dots & C_{1M} \\ R_2 & \rightarrow & C_{21} & C_{22} & \dots & C_{2M} \\ \dots & & & & \dots & \\ R_M & \rightarrow & C_{M1} & C_{M2} & \dots & C_{MM} \end{pmatrix} \begin{matrix} R_i = \max\limits_{r_1 \dots r_M} (t_r + C_{ir}) \\ C_{kn} \in N \cup \{Null\} \end{matrix}$$

Рис. 3. Матрица статической оценки.

Используя это представление, опишем статический этап оценки как перемножение в обратном порядке матриц статической оценки для инструкций (Рис. 4) с модифицированными аддитивной и мультипликативной операциями.

$$E(I_1 \bullet I_2) = E(I_2) \times E(I_1)$$

$$C_1 '+' C_2 = \max(C_1, C_2)$$

$$C_1 '*' C_2 = \begin{cases} C_1 + C_2, если\ C_1, C_2 \neq Null \\ C_1, если\ C_2 = Null \\ C_2, если\ C_1 = Null \end{cases}$$

Рис. 4. Последовательная композиция статических оценок.

Тогда на динамическом этапе оценки исходным состоянием модели процессора является вектор нулей (все ресурсы свободны), а в ходе оценки это состояние умножается на статические оценки для линейных участков в порядке их выполнения (Рис. 5). И снова, аддитивная и мультипликативная операции изменены.

$$S_{i+1} = E(I_i) \times S_i$$

$$C_1 '+' C_2 = \max(C_1, C_2)$$

$$C_1 '*' C_2 = \begin{cases} C_1 + C_2, если\ C_1, C_2 \neq Null \\ 0, если\ C_2 = Null \vee C_1 = Null \end{cases}$$

Рис. 5. Изменение состояния модели процессора.

6. Сложность

Число операций, выполняемых на этапах оценки, можно оценить как

$$L = O(N_{инстр} * N^3_{ресурсов})$$

$$L = O(N_{лин.уч.} * N^2_{ресурсов})$$

Важно отметить, что размер матрицы статической оценки для последовательности инструкций не превосходит квадрата числа ресурсов модели процессора и не зависит от числа инструкций.

7. Заключение

Разработанная нами методика позволяет эффективно оценивать время выполнения ассемблерной программы на вычислителях определенного класса. Используемый математический аппарат позволяет описывать алгоритмы оценки времени выполнения и точно оценивать их сложность.

Методика оценки реализована в составе средства для статико-динамической оценки времени выполнения программ для векторно-скалярного процессора NM6403 NeuroMatrix, где достигнута потактовая (абсолютная) точность оценки; время оценивания для программ, написанных на языке C++, в десятки раз меньше времени эмуляции работы этих программ на инструментальной машине.

Дальнейшая работа может быть направлена на оценку времени выполнения распределенных программ на многопроцессорной целевой машине.

1. Balashov V.V., Kapitonova A.P., Kostenko V.A., Smeliansky R.L., Youshchenko N.V. “Modeling of Digital Signal Processors Based on the Static-Dynamic Approach” (doc) in Proc. of 1st International Conference “Digital signal processing and its applications”, Moscow, MCNTI, vol. IV-E, 1998, p. 79-83.