XXXII Неделя науки СПбГПУ. Материалы межвузовской научно-технической конференции. Ч.V : С.76-78

УДК 004.31

М.И.Глухих (асп., каф. АиВТ), В.Ф.Мелехин, д.т.н., проф.

МЕТОДИКА АНАЛИЗА ЭФФЕКТИВНОСТИ АРХИТЕКТУРЫ ПРОЦЕССОРА

Ограниченность ресурсов на кристалле, обусловленная ограниченными возможностями отечественной технологии производства интегральных схем, определяют **актуальность** поиска наиболее рационального способа использования этих ресурсов, обеспечивающего наибольшую производительность однокристального процессора на заданном классе задач. Цель данной работы – создание методики для оценок эффективности архитектурных решений при разработке специализированного процессора. В связи с многовариантностью результатов синтеза архитектуры актуален сравнительный анализ вариантов, а также выявление и устранение слабых мест архитектуры процессора.

При рассмотрении архитектуры акцент делается на организацию системы команд (безадресная – одноадресная – двухадресная) и на организацию внутрипроцессорной памяти (стековые буферы – регистры общего назначения – аккумуляторы). С учетом известного свойства локальности команд и данных, а также с учетом того, что задержки в связях на кристалле по крайней мере на порядок меньше задержек на печатной плате, основной вопрос при разработке архитектуры – «Как следует организовать внутреннюю память процессора и обращения к ней?».

Также предполагается, что для программирования задач используется язык высокого уровня. Поэтому оценка производительности должна учитывать влияние на нее как архитектуры процессора, так и компилятора с языка высокого уровня.

Оценка влияния архитектуры процессора может производиться несколькими различными способами, которые должны использоваться в совокупности. Первым из них является непосредственное выполнение на вычислительной системе ВС с оцениваемой архитектурой процессора конкретных программ на языке высокого уровня (вместо готовой специализированной ВС может использоваться ее модель – эмулятор на ВМ). Преимущество такого метода – в большой точности получаемой оценки. Однако большое количество программ, которые должны быть написаны и проанализированы, а также длительные затраты на создание эмулятора делают этот метод применимым только на последнем этапе создания системы. Для снижения трудозатрат при разработке имитационной модели вычислительных процессов для системного этапа проектирования предлагается абстрагироваться от деталей архитектуры, имеющих второстепенное значение для производительности, но охватить все этапы организации вычислений. Это – второй способ.

Модель состоит из трех основных частей: генератора программ, компилятора и исполнителя. Генератор формирует последовательности упрощенных программ на языке высокого уровня, которые затем используются для оценки производительности. В программах используются только вычислительные команды типа c = a # b. Прочие команды не учитываются, так как они не имеют прямого отношения к рассматриваемым аспектам архитектуры. Для учета связей по данным программа разбивается на процедуры. Используемые в ней переменные разбиваются на две группы, названные статическими и автоматическими. Статические могут использоваться в любой части программы, а автоматические могут использоваться только в той процедуре, в которой объявлены.

Компилятор осуществляет перевод программы с языка высокого уровня на машинный язык. При переводе команды разрешено анализировать ее и несколько последующих команд (учитывается возможность условных переходов). Исполнитель имитирует выполнение

полученных программ на машинном языке на заданной архитектуре процессора. При задании архитектуры определяется, какие виды внутренней памяти в ней используются, какой они имеют объем, и какие команды в ней разрешены.

Результатом работы имитационной модели является среднее время выполнения команды на языке высокого уровня, а также распределение затрат времени (чтение команд, чтение и запись операндов, внутрипроцессорный обмен, вычислительные команды). Такой метод позволяет получить ответ гораздо быстрее, чем при использовании эмулятора или образца ВС, к тому же, мы абстрагируемся от других деталей архитектуры процессора. Основная трудность здесь состоит в формировании тестовых программ высокого уровня с характеристиками, адекватными реально используемым. И, разумеется, оценка получается не такой точной.

Наконец, третий способ – использование аналитической модели. Такая модель строится аналогично имитационной, однако вместо имитации исполнения команд мы, зная их характеристики и алгоритм работы компилятора, аналитически рассчитываем среднее время их выполнения.

В общем случае эта задача сложна для разрешения. По-видимому, наилучшим методом решения является метод инвариантов. При этом мы описываем состояние внутренней памяти процессора так, чтобы оно осталось неизменным после выполнения следующей команды. Тогда, зная состояние памяти, мы можем определить вероятности нахождения ее операндов в различных участках памяти, составить возможные варианты трансляции на машинный язык и определить среднее время выполнения команды.

Решение данной задачи наиболее просто, когда компилятор использует всю внутреннюю память для хранения определенного набора переменных, причем заранее известно, где какая переменная хранится (например, компилятор сохраняет во внутренней памяти все автоматические переменные). В этом случае мы знаем точно состояние внутренней памяти, и оценить время выполнения команд не составит труда. Если же набор переменных, находящихся во внутренней памяти, постоянно меняется, то приходится использовать вероятностные оценки. В простых случаях это также дает результат, хотя в общем случае такая оценка затруднительна.

Преимуществом аналитической модели является наиболее высокая скорость получения ответа. К тому же, ответ получается не в численном, а в аналитическом виде, что позволяет оценить влияние на производительность различных характеристик программы, задержек тех или иных устройств системы и некоторых других факторов. Минус состоит в том, что аналитическая модель очень узка, по-видимому, не каждый возможный алгоритм компиляции поддается обсчету и не обязательно, чтобы обсчитываемые алгоритмы были наилучшими. Поэтому она требует согласования с имитационной моделью.

Разработанная методика была использована для анализа эффективности специализированного процессора со стековой архитектурой [1] (безадресная архитектура с аппаратным стеком для хранения данных). Было произведено сравнение возможных путей организации компилятора для данной архитектуры [2]. Исследование на имитационной модели показало, что, за редкими исключениями, более эффективным является статичный компилятор, сохраняющий во внутренней памяти постоянный набор переменных. На аналитической модели эти результаты подтвердились.

Производится также сравнение данной архитектуры с архитектурами другого рода, в частности, одноадресной аккумуляторной. До конца сравнение еще не доведено, приведем несколько уже полученных результатов. Стековая архитектура оказывается эффективнее аккумуляторной в случае использования медленной внешней памяти, а также большого количества параллельно используемых переменных. Следует отметить, что для аккумуляторной архитектуры, ввиду ее большей гибкости, эффективнее разделять

внутреннюю память между различными переменными. По-видимому, для большей части традиционно используемых систем аккумуляторная архитектура оказывается удобнее в обработке и быстрее.

В дальнейшем предполагается закончить сравнение рассмотренных архитектур, сравнить их с другими возможными архитектурами (как минимум – регистровой двухадресной), и, по возможности, улучшить архитектуру специализированного процессора [1] с целью повышения его производительности.

ЛИТЕРАТУРА:

1. Глухих М.И., Мелехин В.Ф. Разработка и исследование специализированного процессора для отказоустойчивой системы. XXX юбилейная неделя науки СПбГТУ. Часть VII. Материалы межвузовской научной конференции. С. 152.

2. Глухих М.И., Мелехин В.Ф. Исследование компилятора для стековой архитектуры. Политехнический симпозиум «Молодые ученые – промышленности северо-западного региона». Материалы тематических семинаров по направлениям: «Компьютерные технологии и коммуникации», «Экология и энергоресурсосбережение». 2003. СС. 13-14, 53.