Оптимизация программ за счёт улучшения использования модуля предсказания перехода

Горшков Сергей Сергеевич

Студент

Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия

E-mail: serggorsar@yandex.ru

Научный руководитель *— Сухомлин Владимир Александрович*

С распространением современных конвейерных микропроцессорных архитектур стало возможно сделать независимыми разные шаги выполнения инструкции. Это позволяет сократить время простоя отдельных элементов конвейера и существенно уменьшить время выполнения программы. Но возникают конфликты, которые затормаживают работу конвейера. Например, конфликты по данным — некоторый этап зависит от значения/побочных эффектов, получаемых на предыдущем этапе выполнения инструкции. В работе рассмотрим конфликты по управлению, то есть ветвления/переходы в коде.

В программах часто встречаются блоки кода, которые выполняются в зависимости от выполнения некоторого условия; в случае его невыполнения управление передается другой ветви. Обычно это реализуется с помощью инструкции условного перехода. Пока не будет выяснена истинность условия, неизвестно, какой ветви будет передано управление, поэтому отдельные этапы конвейера будут простаивать. Механизм предсказания ветвлений угадывает, какой ветви будет передано управление, и начинает спекулятивное исполнение инструкций на выбранной ветви. Однако если после вычисления условия выясняется, что предсказание ошибочно, то выполненные инструкции сбрасываются и конвейер запускается с нужной ветвью, что вызывает задержку и может привести к изменению кэшей, блокировкам участка памяти и др. В настоящее время есть немало решений — спекулятивное выполнение обеих ветвей в ожидании результатов вычисления условия, использование статических методов предсказания перехода, опирающихся исключительно на вид инструкций, и динамических, основывающихся на истории выполнения переходов. Последние наиболее распространены и позволяют делать успешные предсказания с вероятностью, превышающей 0.9.

Как правило, программист не задумывается о вышеизложенном, когда пишет код. Некоторые компиляторы очень хорошо могут делать корректные предсказания, но во многих случаях не решают са-

мостоятельно две проблемы, связанные с конфликтами по управлению — предоставление возможности использовать модуль предсказания ветвления (это возможно не для всех инструкций) и уменьшение количества обращений к нему в рамках всей программы. Опишем некоторые из ситуаций, которые оптимизируются в данной работе с помощью методов лексического и синтаксического анализа.

1. Для реализации операторов множественного выбора или таблиц виртуальных функций на уровне ассемблера может создаваться таблица переходов (массив адресов, на которые может быть сделан переход), использующая возможность косвенного перехода/вызова подпрограмм. Для этого случая типовая реализация предсказателя не подходит, поскольку наблюдается множественное ветвление. Делаем преобразование кода, разворачивая операторы множественного выбора в последовательность условных выражений с сохранением fallthrough-переходов и корректной работой с атрибутами (с C++17), используя, возможно, условия, соединенные логическим ИЛИ.

2. В случае условия, состоящего из нескольких выражений, соединенных логическими И/ИЛИ, в языках программирования с реализованными «ленивыми» вычислениями на уровне ассемблера будут условные переходы после каждого из входящих в состав условия выражений, что увеличивает количество однотипных ветвлений. В зависимости от побочных эффектов во второй и последующих частях выражения разделяем вычисления и логические операции над составными частями и используем битовые операции. Падения производительности не будет из-за того, что временные переменные все равно будут положены на регистры при оптимизациях компилятора.

3. Внутри цикла часто встречаются ветвления, не зависящие от состояния программы. В такой ситуации, как правило, будут переходы в конце каждой итерации цикла (проверка условия нахождения в цикле) и ветвление внутри цикла. После надлежащих проверок меняем местами инвариантный переход и цикл, тем самым уменьшая количество обращений к предсказателю во столько раз, сколько ветвлений на каждой итерации.

Литература

1. Fog A. The microarchitecture of Intel, AMD and VIA CPUs. An optimization guide for assembly programmers and compiler makers. Technical University of Denmark, 2018.
2. Jimenez D. A. Fast path-based neural branch prediction // in Proceedings of the 36th annual IEEE/ ACM International Symposium on Microarchitecture, 2003, p. 243.