УДК 004.272.32

Планирование выполнения инструкций для векторных процессоров с переменной длиной векторов

А.Ю. Пантелеев

Национальный исследовательский ядерный университет "МИФИ",

apanteleev87@gmail.com

***Аннотация* — В докладе описано построение блока планировки выполнения инструкций для векторного процессора с переменной длиной векторов. Запуск каждой новой инструкции производится раньше полного завершения предыдущей инструкции. Приведены аналитические выражения для определения времени выполнения тестовых программ в зависимости от длины конвейера и параметров планировщика. Аналитические выражения хорошо согласуются с результатами экспериментов, полученными на RTL-модели векторного процессора. Ранний запуск инструкций позволяет повысить производительность во многих случаях в 1.1–1.9 раза.**

***Ключевые слова* — векторный процессор, переменная длина векторов, планировщик.**

I. Введение

Инструкции типа SIMD (Single Instruction Multiple Data) применяются во многих процессорах и дают возможность получить высокую производительность и энергоэффективность, то есть производительность в расчете на единицу потребляемой мощности [1]. Основная идея этого подхода состоит в том, что каждая инструкция производит не одну, а несколько одинаковых вычислительных операций над элементами векторов определённой длины. Некоторые процессоры, например, Cray X1, предназначенный для суперкомпьютеров, применяют особый вариант SIMD: вектора с переменной длиной [2]. Это позволяет сделать программы более универсальными, уменьшить количество циклов в программах и, как следствие, зависимостей по управлению (control dependencies). Реализовать такой подход в простейшем случае несложно: достаточно разделить длинные вектора на короткие *векторные элементы* и выполнять их последовательно; каждая следующая инструкция начинает выполняться лишь тогда, когда предыдущая завершается (т.е. записывает в регистры все результаты своей работы). Если длина векторов задается с точностью до одного скалярного элемента, то программа может без модификаций работать на векторных процессорах с различным количеством параллельных вычислительных блоков.

Если последовательные инструкции используют один вычислительный блок, то для повышения производительности следует начинать выполнять следующую инструкцию тогда, когда готовы данные для первого векторного элемента, а не тогда, когда завершается вся предыдущая инструкция (далее такое поведение называется *ранним запуском*). Это особенно актуально при большой длине конвейера и при малой длине векторов. Диаграмма использования абстрактного конвейера из 5 стадий для обоих подходов представлена на рис. 1: изображено выполнение двух зависимых инструкций, каждая из которых обрабатывает по 3 векторных элемента.

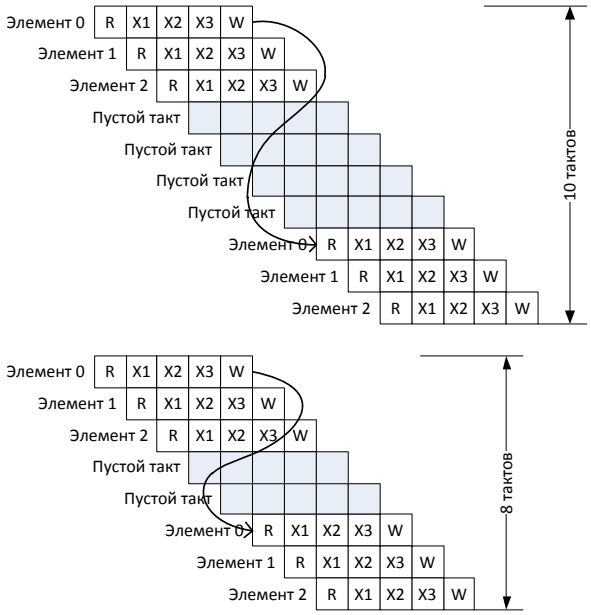

Рис. 1. Сокращение времени выполнения программы при запуске следующей векторной инструкции до полного завершения предыдущей

Построение такого планировщика в целом также несложно: достаточно начинать выполнять новую инструкцию тогда, когда записаны результаты работы первого векторного элемента предыдущей инструкции (или когда последний векторный элемент предыдущей инструкции отправлен в конвейер, если это произошло позже). Задача усложняется, если обработка одного векторного элемента от разных инструкций может занимать разное время – это может

быть связано с различной сложностью вычислений или с чтением операндов для этих инструкций. Далее предлагается решение этой задачи, основанное на расчете времени ожидания для каждой пары зависимых инструкций.

II. Предлагаемое решение задачи планирования

Рассмотрим реализацию планировщика для векторного процессора со следующими особенностями архитектуры:

1. Длина векторов задаётся один раз для группы выполняемых инструкций и обозначается как VL (сокращение от Vector Length), *измеряется в векторных элементах*.

2. Имеется один многофункциональный векторный вычислительный блок, который может выполнять инструкции за один или два такта. Глубина конвейера вычислительного блока – D (Depth) стадий. Количество параллельных вычислительных устройств для планировщика значения не имеет, т.к. он работает с векторными элементами.

3. Инструкции могут иметь от одного до трёх операндов и производить один или два результата. Чтение операндов и передача их в вычислительный блок может занимать от 1 до 3 тактов, в зависимости от структурных конфликтов в системе векторной памяти.

Структурная схема планировщика и окружающих блоков показана на рис. 2. Планирование выполнения инструкции начинается с того момента, когда декодированная инструкция поступает на вход планировщика. У инструкции имеются следующие поля: код операции, три имени регистров-операндов и два имени регистров-результатов. Каждое имя регистра имеет флаг присутствия, информацию о странице памяти, в которой находится регистр, и номер регистра в странице.

A. *Составление плана*

На первой стадии своего конвейера планировщик составляет план обработки векторных элементов. План – это структура данных, содержащая следующие поля:

- два номера позиции операндов для чтения на первом такте – например, «считать первый и третий операнды»;
- по одному номеру позиции операндов для чтения на втором и третьем тактах;
- по одному номеру позиции результатов для записи на первом, втором и третьем тактах;
- количество тактов, необходимое для обработки одного векторного элемента (значение от 1 до 3) – обозначается как CPE (Cycles Per Element).

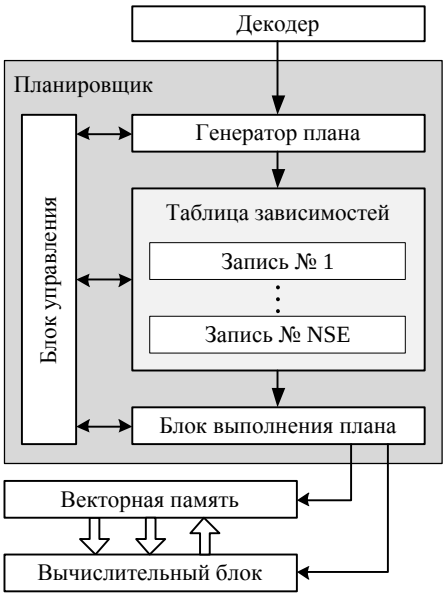

Рис. 2. Структура планировщика

План строится на основании данных о размещении регистров в страницах векторной памяти и гарантирует отсутствие структурных конфликтов (т.е. двух обращений к одной странице памяти за такт). Реализация блока составления планов – табличная логическая функция.

B. *Таблица зависимостей*

Составленный план поступает в таблицу зависимостей (Scoreboard). Эта таблица решает основную задачу: определить момент, в который можно начинать выполнять следующую инструкцию. Для этого в таблице сохраняется несколько записей об активных инструкциях; обозначим их максимальное количество как NSE (Number of Scoreboard Entries). Выбор значения NSE оказывает существенное влияние на производительность процессора: если это значение слишком мало, то процессор не сможет эффективно выполнять поток независимых инструкций, работающих с короткими векторами. В разделе III приведены аналитические выражения, позволяющие оценить производительность в зависимости от различных факторов, включая NSE – эти выражения можно использовать для выбора оптимальных значений параметров планировщика.

Каждая запись в таблице является конечным автоматом с 4 состояниями: EMPTY, ALLOCATED, STARTED и BLOCKING. Диаграмма состояний автомата показана на рис. 3. Изначально каждая запись находится в состоянии EMPTY (пусто). После генерации плана для инструкции выделяется свободная запись (если таковая есть), которая переходит в состояние ALLOCATED. Как только инструкция запускается на выполнение, запись переходит в состояние STARTED и по завершению выполнения – обратно в EMPTY. Переход в состояние

EMPTY производится при достижении нулевого значения счетчиком TS (Time to Stop), который инициализируется при запуске инструкции и на каждом последующем такте уменьшает свое значение на единицу. Начальное значение счетчика определяется по следующей формуле:

$$TS = CPE * VL + D. \quad (1)$$

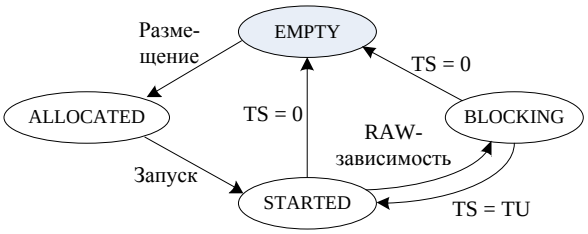

Рис. 3. Диаграмма состояний автоматов каждой записи в таблице зависимостей

То есть, время выполнения инструкции состоит из времени выпуска (повторение плана для каждого векторного элемента) и времени прохода через конвейер последнего векторного элемента (латентности D).

При размещении новой инструкции в определённой записи (назовем эту инструкцию NEW) всем остальным записям даётся команда проверить наличие RAW-зависимостей (чтение после записи – Read After Write hazard). Если любая другая запись, содержащая активную инструкцию (назовем ее OLD), обнаруживает зависимость, то она переходит из состояния STARTED в состояние BLOCKING, то есть блокирует выполнение инструкции NEW. Время разблокировки рассчитывается заранее. Удобнее всего измерять это время не от момента проверки зависимостей, а от момента завершения инструкции OLD. Момент запуска зависимой инструкции NEW выбирается так, чтобы все её векторные элементы получили корректные данные без изменения плана. То есть, если значение CPE инструкции NEW больше или равно значению CPE инструкции OLD, то инструкцию NEW можно запускать сразу после того, как будет записан первый векторный элемент результата инструкции OLD. Если же значение CPE инструкции NEW меньше, чем у инструкции OLD, то запуск нужно отложить на такое время, чтобы последний векторный элемент инструкции NEW получил корректные данные. Используется следующая формула для расчёта значения счётчика TS, при котором следует перевести инструкцию OLD обратно в состояние STARTED (обозначается TU от Time to Unblock):

$$TU = \min (NEW: CPE, OLD: CPE) * (VL - 1). \quad (2)$$

Иллюстрация к формулам (1) и (2) приведена на рис. 4. Изображена обработка инструкции OLD при длине вектора VL = 3 и CPE = 2; также изображены два варианта обработки инструкции NEW: со значением CPE = 3 и CPE = 1. Чтение операндов для инструкций показано сплошными линиями, а запись результатов инструкции OLD – пунктирными. В первом случае (при CPE = 3) время запуска инструкции NEW определяется первым векторным элементом, что показано прямоугольником; во втором случае время запуска определяется последним векторным элементом.

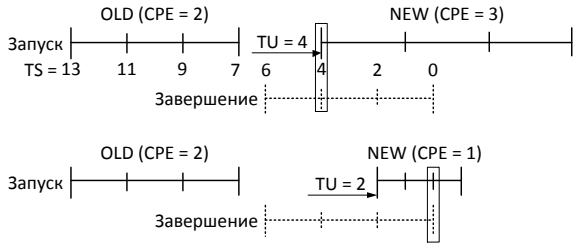

Рис. 4. Пример определения времени запуска зависимой инструкции

C. Выполнение

Когда новая инструкция размещена в таблице, ни одна из активных инструкций не препятствует выполнению, и все активные инструкции уже полностью отправлены в конвейер, новая инструкция начинает выполняться. При наличии сгенерированного плана выполнение инструкции сводится к повторению этого плана для каждого векторного элемента и не составляет сложностей.

III. Производительность планировщика

Проведем оценку производительности планировщика при помощи серии тестов. Каждый тест состоит из NI (Number of Instructions) независимых последовательностей инструкций. Инструкции, принадлежащие разным последовательностям, чередуются; для NI = 3 программа состоит из одинаковых фрагментов, содержащих 3 независимых инструкции:

a = a + const; b = b + const; c = c + const;

a = a + const; b = b + const; c = c + const; …

Производительность может быть ограничена различными факторами, для каждого из которых можно построить теоретическое предсказание времени выполнения последовательности из NI инструкций, предполагая большое количество повторений такого блока. Результат одного или нескольких из выражений (4–7), имеющий максимальное значение, определяет производительность для каждого конкретного случая:

$$T_{Overall} = \max (T_{Issue}, T_{Latency}, T_{Control}, T_{Scoreboard}). \quad (3)$$

A. Пропускная способность внешних устройств

Случай, когда производительность процессора ограничена пропускной способностью регистровой памяти или вычислительного устройства, можно

назвать идеальным для планировщика. Это означает, что в программе достаточно параллелизма, чтобы загрузить вычислительный блок работой. В этом случае время обработки группы из NI инструкций определяется только исходя из запланированного числа тактов на элемент:

$$T_{Issue} = CPE * VL * NI. \quad (4)$$

B. Латентность инструкций

Если в программе недостаточно параллелизма, чтобы покрыть латентность вычислений, то производительность программы ограничена латентностью. Простейшим примером такого случая может быть последовательность зависимых инструкций (NI = 1), состоящих из очень коротких векторов (VL = 1). Время обработки группы инструкций в этом случае определяется по следующей формуле:

$$T_{Latency} = CPE + D. \quad (5)$$

C. Количество записей в таблице зависимостей

Если в программе имеется большое количество независимых последовательностей инструкций, работающих над относительно короткими векторами, то таблица зависимостей может заполниться и прекратить принимать новые инструкции, пока не завершится хотя бы одна активная инструкция. Формула (6) описывает среднее время выполнения группы из NI инструкций.

$$T_{Scoreboard} \cong (CPE * VL + D + 1) * NI / NSE. \quad (6)$$

D. Производительность управляющего автомата

Конечный автомат, который управляет продвижением инструкции от генератора плана в таблицу зависимостей и далее в блок выполнения плана, может обрабатывать только одну инструкцию каждые три такта (обозначим это значение RL – от Reissue Latency). При очень маленьких векторах и низкой латентности это может являться ограничением производительности. В этом случае время обработки группы инструкций определяется по следующей формуле:

$$T_{Control} = RL * NI. \quad (7)$$

E. Результаты экспериментов

Эксперименты, проведенные на RTL-модели процессора с описанным планировщиком, показывают производительность, которая в 97% случаев отличается от теоретической не более чем на 3%. На рис. 5 приведены графики зависимости реального количества векторных элементов, обработанных за такт, от длины векторов VL для различных значений NI. Значения других параметров: D = 14, NSE = 6, CPE = 1, RL = 3. На рис. 6 представлены графики ускорения выполнения тех же программ при переходе от запуска инструкции только после завершения предыдущей к раннему запуску.

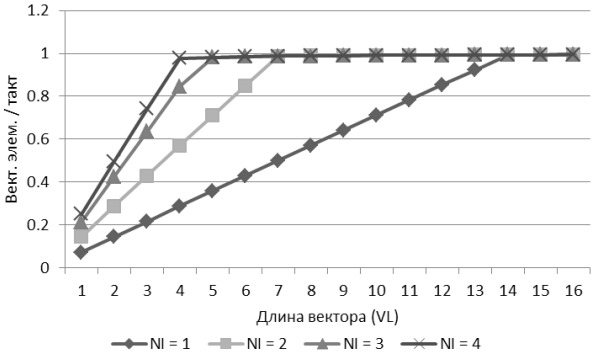

Рис. 5. Зависимость числа обработанных векторных элементов за такт от VL

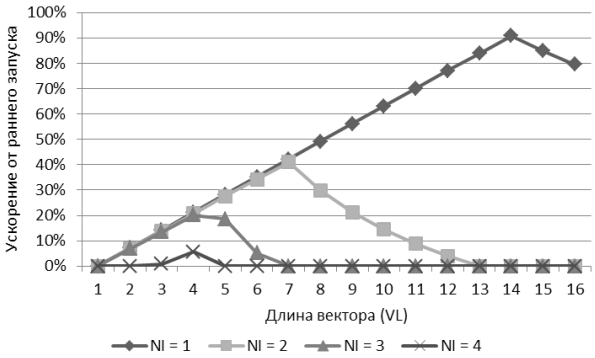

Рис. 6. Влияние раннего запуска инструкций на производительность программы в зависимости от VL

IV. Заключение

Представленный подход может применяться при проектировании векторных процессоров с переменной длиной векторов и длинным конвейером, предназначенных для быстрой обработки небольших вычислительных задач (например, при цифровой обработке сигналов). Ранний запуск инструкций позволяет повысить производительность программ на 10–90%, в зависимости от длины векторов и от количества параллелизма, доступного программе на уровне инструкций (ILP).

Литература

[1] Kozyrakis C., Patterson D. Scalable Vector Processors for Embedded Systems // Micro, IEEE. 2003. V. 23. I. 6. pp. 36 – 45.

[2] Hennessy J., Patterson D. Computer Architectute: a Quantitative Approach – 4th ed., Appendix F // США: Morgan Kaufmann, 2007.