Вестник Приднестровского университета. Тирасполь, 1995. № 1. С. 124-127.

В.С. Выхованец, К.П. Гордиенко

ПРОЦЕССОР С СОКРАЩЕННЫМ НАБОРОМ КОМАНД

Основным противоречием в развитии архитектуры процессоров с сокращенным набором команд является несоответствие скорости поступления команд и данных в процессор и скорости их обработки Согласование быстродействия процессора с быстродействием внешних запоминающих устройств возможно вследствие фундаментального свойства локальности потоков команд и данных, заключающегося в слабой зависимости процесса обработки от команд и данных, которые расположены на большом расстоянии друг от друга как во времени, так и в пространстве.

Одним из способов достижения равномерности потоков команд и данных при существующих ограничениях пропускной способности внешней шины процессора является использование кэш-памяти. Эффект от ее применения появляется только в том случае, если удается минимизировать затраты времени на ее перезагрузку при ветвлении в потоке команд и при ветвлении в потоке данных. В [1] предложен способ избежать потерь в быстродействии из-за ветвлений путем разделения кэш-памяти команд на две части с возможностью параллельной работы каждой из них. В [2] описывается метод прогнозирования ветвлений, позволяющий производить опережающее заполнение кэш-памяти.

Другим способом использования локальности потоков команд и данных и способом повышения быстродействия процессора является усложнение

функций обработки данных VLIW-архитектура [3] позволяет в одной команде закодировать большое число простых операций и запланировать операции, расположенные как до, так и после условного перехода В [2] рассмотрен принцип суперскалярной обработки данных, заключающийся в параллельном выполнении двух команд с разрешением зависимостей между ними по данным, ресурсам и управляющим признакам.

Равномерность потоков команд и данных может быть также улучшена реконфигурацией процессора через изменение набора команд [4], реализацией мультипрограммного режима обработки путем контекстных переключений процессора [5] и использованием стековой организации доступа к данным [6].

Все перечисленные способы обеспечения равномерности потоков команд и данных и повышения быстродействия процессора прямо или косвенно основываются на их свойстве локальности и учитывают лишь отдельные особенности процесса обработки, которые навязываются используемыми парадигмами программирования [7].

Существенного повышения быстродействия обработки данных можно достичь лишь при комплексном учете свойства локальности потоков команд и данных в рамках новых нетрадиционных архитектурных принципов построения процессоров, ориентированных на поддержку локальной парадигмы программирования, которая позволяет сформировать эффективные входные и выходные потоки процессора.

Предлагаются следующие архитектурные принципы построения процессоров с сокращенным набором команд путем ориентации на новую Локальную парадигму программирования.

1. Организация кэш-памяти данных процессора в виде стека - для доступа к операндам и одновременно в виде очереди - для подкачки данных при отрицательном и сохранения данных при положительном переполнениях стека Доступ к операндам возможен не только на вершине стека данных, но и на глубине, не превышающей объема кэш-памяти.

2. Организация кэш-памяти команд в виде нескольких очередей (секций) с опережающей выборкой команд из внешней памяти таким образом, чтобы во время выполнения текущей команды в кэш-памяти находились все команды, расположенные до и после текущей на заданном расстоянии.

3. Выборка за один цикл шины процессора одновременно нескольких команд, что достигается путем сокращения длины команды и, следовательно, общего количества аппаратно-реализуемых команд и увеличением разрядности шины данных.

4. Разделение команд на два типа: на команды нижнего уровня, непосредственно выполняемые процессором, и команды верхнего уровня, каждая из которых состоит из последовательности команд нижнего или верхнего уровней, завершающейся командой возврата из команды верхнего уровня. Параллельно с подкачкой команд в текущую секцию кэш-памяти команд производится опережающее заполнение свободных секций последовательностью команд, составляющих команду верхнего уровня, которая встретилась при опережающей подкачке команд в текущую секцию.

5. Использование стека команд для хранения данных о состоянии секций кэш-памяти команд. Основной цикл процессора заключается в последовательной выборке команд из текущей секции кэш-памяти, определении принадлежности команды к типу команд нижнего или верхнего уровня с последующим выполнением команды нижнего уровня или выборкой

кода следующей команды из соответствующей секции при сохранении состояния текущей секции в стеке команд. Команда возврата из команды верхнего уровня возвращает процессор к выполнению команды из секции, данные о которой находятся на вершине стека команд. Подкачка данных при отрицательном и сохранение данных при положительном переполнениях стека команд вызывает опережающую подкачку в свободную или освободившуюся секцию соответствующих команд.

6. Выделение специальных секций кэш-памяти команд для хранения определений наиболее часто используемых команд верхнего уровня, что приводит к их длительному или постоянному присутствию в кэш-памяти. На этапе компиляции компилятором определяются для каждой текущей локальной области кода наиболее часто используемые команды верхнего уровня, и в ее начале генерируется код опережающей подкачки в специальные секции определений соответствующих команд.

7. Разделение кэш-памяти данных и команд процессора на несколько контекстных областей по одной для каждого процесса. При контекстном переключении, выполняемом за один такт, происходит переключение соответствующих областей кэш-памяти, что обеспечивает минимально возможное время переключения контекста при автоматическом сохранении не только содержимого кэш-памяти данных, но и содержимого соответствующих секций кэш-памяти команд и стека команд текущего процесса.

8. Ввод в состав процессора нескольких операционных устройств, в том числе и требующих более чем один такт для обработки данных. Занятость операционного устройства, разделяемого между процессами или на уровне команд вызывает контекстное переключение процессора. Каждое разделяемое устройство, в том числе и построенное по конвейерному принципу, по завершении обработки данных вызывает контекстное переключение на процесс, ожидающий результаты. При этом у компилятора появляется возможность генерации кода параллельных вычислений как на уровне процессов, так и на уровне команд нижнего уровня. Для увеличения быстродействия процессора путем параллельной обработки данных возможно использование и нескольких однотипных операционных устройств. Устройство связи с внешней шиной содержит очередь запросов, что позволяет выполнить операцию записи во внешнюю память за один такт, а при выполнении операции чтения произвести контекстное переключение на другой процесс до завершения этой операции.

9. Придание процессору открытой архитектуры, которая обеспечивает его дальнейшее развитие и предполагает переменный или изменяемый состав его операционных устройств. Система команд содержит поле, определяющее адрес устройства, выполняющего операцию, и поле кода операции для этого устройства Базовое программное обеспечение идентифицирует состав и типы устройств по слову конфигурации процессора и доопределяет систему команд, если требуемое устройство не входит в его состав.

Для проверки предложенных архитектурных принципов проведено моделирование процессора с сокращенным набором команд на ЭВМ с использованием следующих параметров: длина команды – 2 байта; разрядность шины данных процессора - 8 байт; количество секций кэш-памяти команд – 32; объем одной секции – 256 байт, объем кэш-памяти данных – 256 байт; время цикла внешней шины – 2 такта, количество поддерживаемых процессов – 16. Результаты моделирования процессора показали его эффективность.

Вестник Приднестровского университета. Тирасполь, 1995. № 1. С. 124-127.

Цитированная литература

1. Пат. 5050076 США Заявлено 04.10.88. Опубликовано **17.09.91.**

2 **Alpert D., Avnon D.** Architecture of pentium microprocessor // IEEE Micro. 1993. Vol. 13, № 3. **P. 11-21.**

3. **Дризовский Л.М., Меньшикова Л.А., Тинина Н.В., Кулаков А.Г.** Современное состояние и тенденции развития супермини-ЭВМ // Приборы, системы автоматизации и системы управления. ТС-2 «Средства вычислительной техники и оргтехники» 1990. Вып. 1. С. 21-22.

4. **Athanas Peter M., Selverman Harvey F.** Processor reconfiguration through instruction-set metamorphosis // Computer. 1993. Vol. 26, № 3. С. 11-18.

5 **Agarwel A., Kubiatowicz J., Kranz D. et** al. Sparcle: An evolutionary
processor design for large scale multiprocessors // IEEE Micro. 1993 Vol. 13, № 3. P. 48-61.

6 **Харько В.В, Герасько В.В, Галицкий А.В.** Бортовые ЭВМ языков высокого уровня: Проблемы повышения производительности // Электронная техника. Сер. 10. 1992, № 1-2. С 3-6

7. **Ambler A. L., Burnett M. M., Zimberman B. A.** Operation versus definitional: A perspectiv on programming paradigms // Computer. 1992 Vol. 25, № 9. С 28-43.