Мнение: Как я вижу современный процессор
Ответ на статью "Создавая новое поколение - часть 1"
batu (Миша Кузьмин), Четверг, 12 Август 2004, 15:48

Естественно, многопроцессорня работа, естественно диспетчер процессоров и памяти. Стековая система команд наиболее подходит к такой архитектуре. Так как алгоритмы распаралеливани выполнения команд проще и проще алгоритмы возможности кэширования. От двухадресных команд как сейчас толку мало. И обязательно апаратная реализация проверки типов (для этого специальное представление данных а не просто байты), и вызов процедур и функций тоже реализация аппаратная. Естественно, тут же работа для диспетчера памяти... В таком виде это будет шикарная молотилка. Кстати, у меня все это расписано. Мне кажется, что это очевидные вещи... А теперь подробнее.

То что многопроцессорне системы нужны - это очевидно. В таком виде как это делается сейчас, это бессмысленно. Распараллеливане работы процессора должно происходить более эффективно и независимо (скрыто от пользователя). Кстати, я против морочить голову программисту потоками. Не хватает ресурсов для решения задачи - покупай более дорогую машину или добавляй процессор. Программистскиеухищрения не приводят к серьезным положительным результатам. Гораздо эффективней должны работать алгоритмы распараллеливаня. Отсюда следует необходимость диспетчера процессоров и, как следствие, диспетчер памяти для распределения памяти между процессорами и процессами. Сборка мусора, кэширование, распределение памяти между задачами и выделение памяти для вызова процедур - вполне решаемая и современная задача для апаратного диспетчера памяти. Тем более, в том или ином виде он уже давно присутствует в современных процессорах.

Еще одна серьезная задача – эффективная диспетчеризация процессоров. Она неразрывно связана с диспетчером памяти, блокировкой выполнения команд, требующих данных, которые изменяются другим процессором, синхронизация и прочие, согласитесь, уже очевидные, но реально выполнимые задачи.. Не надо только их усложнять двухадресными командами... Для этого гораздо лучше подходят стековые команды... Они же, в сочетании с активным кэшированием и многоуровневой памятью, будут работать эффективнее двухадресных.

Еще одна перезревшая задача. Представление данных в компьютере устарело. Тогда как современное программировани давно сориентировано на объектное программировани, проверка типов данных аппаратным способом во время выполнения просто необходима. Возможность проверять данные на типы, ограничения, события и т.п. выпоняемые параллельно с обработкой настолько ускорит выполнение программ, и не просто ускорит выполнение, но и сделает ПО значительно надежнее. Такая проверка все равно делается и требует массу ресурсов. Решение этой задачи аппаратно значительно упростит систему команд, но потребует совершенно новых способов работы процессора, в связи пользовательскии типами и перегрузкой операторов. Создание пользовательски типов, пространств имен, сборка типов в библиотеки потребуют более тесной работы с процессором. Но это вполне решаемая задача. Хотя это и необычно для существующих архитектур.

Здесь же хочется сказать что вытекает из такого подхода для програмистов.

Во-первых, забываем про классы. Это все типы теперь. И определение класса в терминологии .Net - это просто определение типа объекта и ничего больше. Что значительно проще и логичнее.

Второе. Определение коллекций и массивов как классов не есть логично. Гораздо проще иметь такое понятие как организацию данных. А работа с типами данных Object и Integer не должны отличаться ничем. Именно к этому и приходит .Net, только почему-то осторожно. Подробно на этих моментах в данной статье я останавливаться не буду. Это отдельная и непростая тема. Могу сказать только, что получается гораздо проще и логичнее чем в .Net. А работу с организованными данными можно тоже переложить на процессор. Могу только сказать, к организации данных относятся не только массивы и коллекции, но и стеки, списки, кольца и деревья.

Перекладывание работы с типами данных на процессор возлагает еще одну естественную задачу на процессор. Обеспечение работы со свойствами, методами и событиями. При таком подходе события, свойства и методы могут иметь любые данные, даже Integer (заметим, что добавил еще такое понятие как ограничение. Что это такое не буду останавливаться здесь, только замечу, что перечислимые типы и данные можно определять как данные с ограничением на принимаемые значения. Хотя общая формулировка ограничений гораздо шире). Необходима стандартизация работы. Методы, свойства, события и ограничения в терминологии .Net необходимо объединить в слово "" типа метод, свойство, событие и ограничение. Все это реально. И получается очень удобная объектно-ориентированная реально многопроцессорня машина.

Понятно что такой подход требует известной свободы мышления. Но в итоге получается гораздо логичнее, проще и надежнее в работе. Конечно, это принципиальные изменения архитектуры когда нет данных как просто байтов. А есть только данные с типами... А потоки, конвейеры и прочее - это подходы уже 30 летней давности. Пора идти дальше.

Миша Кузьмин aka batu.