https://www.ixbt.com/cpu/ia64.html

16 ФЕВРАЛЯ 1998 Г. НИКОЛАЙ ДОРОФЕЕВ
Архитектура IA64

Для компаний Intel и Hewlett-Packard не существует "проблемы 2000 года" — для них это год новых возможностей. В конце 1999 года Intel планирует представить Merced — первый процессор, построенный с использованием архитектуры нового поколения, совместно разработанной двумя компаниями. Хотя эта 64-разрядная архитектура основана на многолетних исследованиях Intel, HP, других компаний и университетов, она радикально отличается от всего, что было до сих пор представлено на рынке. Достигнет эта архитектура успеха или нет, одно очевидно уже сейчас: она изменит всю компьютерную индустрию.

Эта архитектура, известная под названием Intel Architecture-64 (IA-64), полностью "порывает с прошлым". IA-64 не является как 64-разрядным расширением 32-разрядной архитектуры x86 компании Intel, так и переработкой 64-разрядной архитектуры PA-RISC компании HP. IA-64 представляет собой нечто абсолютно новое — передовую архитектуру, использующую длинные слова команд (long instruction words — LIW), предикаты команд (instruction predication), устранение ветвлений (branch elimination), предварительную загрузку данных (speculative loading) и другие ухищрения для того, чтобы "извлечь больше параллелизма" из кода программ.

Несмотря на то, что Intel и HP обещали добиться обратной совместимости с существующим программным обеспечением, работающим на процессорах архитектур x86 и PA-RISC, они до сих пор не разглашают, каким образом это будет сделано. На самом деле обеспечить такую совместимость совсем не просто; достаточно вспомнить гораздо менее кардинальный переход с 16-разрядной на 32-разрядную архитектуру x86, продолжавшийся 12 лет и до сих пор не завершённый.

По поводу совместимости, стоит заметить, что но в Merced на самом деле существует два режима декодирования команд VLIW и старый CISC. Т.е. программы переключаются в необходимый режим исполнения. В архитектуре x86 были добавлен ряд команд для перехода в новый режим, а также для передачи данных. В IA-64 такие команды есть изначально. Так что теперь ОС будут содержать и 64-х разрядную часть на IA-64 и старую 32-х разрядную.

Правда, переход к архитектуре IA-64 в ближайшее время вряд ли затронет большинство пользователей, поскольку Intel заявила, что Merced разрабатывается для серверов и рабочих станций класса high-end, а не для компьютеров среднего уровня. Фактически, компания заявила, что IA-64 не заменит x86 в ближайшем будущем. Похоже на то, что Intel и другие поставщики продолжат разрабатывать чипы x86.

Перед тем, как углубиться в технические детали, попробуем понять, почему Intel и HP рискнули пойти на столь кардинальные перемены. Причина сводится к следующему: они считают, что как CISC, так и RISC-архитектуры исчерпали себя.

Небольшой экскурс в прошлое. Архитектура x86 компании Intel — CISC архитектура, появившаяся в 1978 году. В те времена процессоры представляли собой скалярные устройства (то есть могли в каждый момент времени выполнять только одну команду), при этом конвейеров практически не было. Процессоры содержали десятки тысяч транзисторов. PA-RISC компании HP была разработана в 1986 году, когда технология суперскалярных (с возможностью выполнения нескольких команд одновременно) конвейеров только начала развиваться. Процессоры содержали сотни тысяч транзисторов. В конце 90-х наиболее совершенные процессоры содержат миллионы транзисторов. К моменту начала выпуска Merced компания Intel планирует перейти на 0.18-микронную технологию вместо нынешней 0.25-микронной. Уже первые чипы архитектуры IA-64 будут содержать десятки миллионов транзисторов. В дальнейших модификациях их число увеличится до сотен миллионов.

Разработчики процессоров стремятся создавать чипы, содержащие как можно больше функциональных узлов — что позволяет обрабатывать больше команд параллельно — но одновременно приходится существенно усложнять управляющие цепи для распределения потока команд по обрабатывающим узлам. На данный момент лучшие процессоры не могут выполнять более четырёх команд одновременно, при этом управляющая логика занимает слишком много места на кристалле.

В то же время, последовательная структура кода программ и большая частота ветвлений делают задачу распределения потока команд крайне сложной. Современные процессоры содержат огромное количество управляющих элементов для того, чтобы минимизировать потери производительности, связанные с ветвлениями, и извлечь как можно больше "скрытого параллелизма" из кода программ. Они изменяют порядок команд во время исполнения программы, пытаются предсказать, куда необходимо будет перейти в результате очередного ветвления, и выполняют команды до вычисления условий ветвления. Если путь

ветвления предсказан неверно, процессор должен сбросить полученные результаты, очистить конвейеры и загрузить нужные команды, что требует достаточно большого числа тактов. Таким образом, процессор, теоретически выполняющий четыре команды за такт, на деле выполняет менее двух.

Проблему ещё осложняет тот факт, что микросхемы памяти не успевают за тактовой частотой процессоров. Когда Intel разработала архитектуру x86, процессор мог извлекать данные из памяти с такой же скоростью, с какой он их обрабатывал. Сегодня процессор тратит сотни тактов на ожидание загрузки данных из памяти, даже несмотря на наличие большой и быстрой кэш-памяти.

Говоря о том, что CISC- и RISC-архитектуры исчерпали себя, Intel и HP имеют в виду обе эти проблемы. В двух пространных интервью журналу BYTE они раскрыли некоторые детали архитектуры IA-64.

- Команды в формате IA-64 упакованы по три в 128-битный пакет для быстрейшей обработки. Обычно это называют "LIW encoding". (Русский аналог подобрать сложно. Наиболее адекватно, на мой взгляд, перевести как "кодирование в длинные слова команд".) Однако компания Intel избегает такого названия, заявляя, что с ним связаны "негативные ассоциации" (negative connotation). По той же причине Intel не любит называть сами команды RISC-подобными (RISC-like), даже несмотря на то, что они имеют фиксированную длину и предположительно оптимизированы для исполнения за один такт в ядре, не нуждающемся в микрокоде. Intel предпочитает называть свою новую LIW-технологию Explicitly Parallel Instruction Computing или EPIC (Вычисления с Явной Параллельностью Инструкций, где "явной" означае явно указанной при трансляции). В любом случае формат команд IA-64 не имеет ничего общего с x86. Команды x86 могут иметь длину от 8 до 108 бит, и процессор должен последовательно декодировать каждую команду после определения её границ.

- Каждый 128-битный пакет содержит шаблон (template) длиной в несколько бит, помещаемый в него компилятором, который указывает процессору, какие из команд могут выполняться параллельно. Теперь процессору не нужно будет анализировать поток команд в процессе выполнения для выявления "скрытого параллелизма". Вместо этого наличие параллелизма определяет компилятор и помещает информацию в код программы. Каждая команда (как для целочисленных вычислений, так и для вычислений с плавающей точкой) содержит три 7-битных поля регистра общего назначения (РОН). Из этого следует, что процессоры архитектуры IA-64 содержат 128 целочисленных РОН и 128 регистров для вычислений с плавающей точкой. Все они доступны программисту и являются регистрами с произвольным доступом (programmer-visible random-access registers). По сравнению с процессорами x86, у которых всего восемь целочисленных РОН и стек глубины 8 для вычислений с плавающей точкой, IA-64 намного "шире" и, соответственно, будет намного реже простаивать из-за "нехватки регистров".

- Компиляторы для IA-64 будут использовать технологию "отмеченных команд" (predication) для устранения потерь производительности из-за неправильно предсказанных переходов и необходимости пропуска участков кода после ветвлений. Когда процессор встречает "отмеченное" ветвление в процессе выполнения программы, он начинает одновременно выполнять все ветви. После того, как будет определена "истинная" ветвь, процессор сохраняет необходимые результаты и сбрасывает остальные.

- Компиляторы для IA-64 будут также просматривать исходный код с целью поиска команд, использующих данные из памяти. Найдя такую команду, они будут добавлять пару команд - команду предварительной загрузки (speculative loading) и проверки загрузки (speculative check). Во время выполнения программы первая из команд загружает данные в память до того, как они понадобятся программе. Вторая команда проверяет, успешно ли произошла загрузка, перед тем, как разрешить программе использовать эти данные. Предварительная загрузка позволяет уменьшить потери производительности из-за задержек при доступе к памяти, а также повысить параллелизм.

Из всего вышесказанного следует, что компиляторы для процессоров архитектуры IA-64 должны быть намного "умнее" и лучше знать микроархитектуру процессора, код для которого они вырабатывают. Существующие чипы, в том числе и RISC-процессоры, производят гораздо больше оптимизации на этапе выполнения программ, даже при использовании оптимизирующих компиляторов. IA-64 перекладывает практически всю работу по оптимизации потока команд на компилятор. Таким образом, программы, скомпилированные для одного поколения процессоров архитектуры IA-64, на процессорах следующего поколения без перекомпиляции могут выполняться неэффективно. Это ставит перед поставщиками нелёгкую задачу по выпуску нескольких версий исполняемых файлов для достижения максимальной производительности.

Другим не очень приятным следствием будет увеличение размеров кода, так как команды IA-64 длиннее, чем 32-битные RISC-команды (порядка 40 бит). Компиляция при этом будет занимать больше времени, поскольку IA-64, как уже было сказано, требует от компилятора гораздо больше действий. Intel и HP заявили, что уже работают совместно с поставщиками средств разработки над переработкой этих программных продуктов.

Технология "отмеченных команд" является наиболее характерным примером "дополнительной ноши", перекладываемой на компиляторы. Эта технология является центральной для устранения ветвлений и управления параллельным выполнением команд.

Обычно компилятор транслирует оператор ветвления (например, IF-THEN-ELSE) в блоки машинного кода, расположенные последовательно в потоке. В зависимости от условий ветвления процессор выполняет один из этих блоков и перескакивает через остальные. Современные процессоры стараются предсказать результат вычисления условий ветвления и предварительно выполняют предсказанный блок. При этом в случае ошибки много тактов тратится впустую. Сами блоки зачастую весьма малы — две или три команды, — а ветвления встречаются в коде в среднем каждые шесть команд. Такая структура кода делает крайне сложным его параллельное выполнение.

Когда компилятор для IA-64 находит оператор ветвления в исходном коде, он исследует ветвление, определяя, стоит ли его "отмечать". Если такое решение принято, компилятор помечает все команды, относящиеся к одному пути ветвления, уникальным идентификатором, называемым предикатом (predicate). Например, путь, соответствующий значению условия ветвления TRUE, помечается предикатом P1, а каждая команда пути, соответствующего значению условия ветвления FALSE — предикатом P2. Система команд IA-64 определяет для каждой команды 6-битное поле для хранения этого предиката. Таким образом, одновременно могут быть использованы 64 различных предиката. После того, как команды "отмечены", компилятор определяет, какие из них могут выполняться параллельно. Это опять требует от компилятора знания архитектуры конкретного процессора, поскольку различные чипы архитектуры IA-64 могут иметь различное число и тип функциональных узлов. Кроме того, компилятор, естественно, должен учитывать зависимости в данных (две команды, одна из которых использует результат другой, не могут выполняться параллельно). Поскольку каждый путь ветвления заведомо не зависит от других, какое-то "количество параллелизма" почти всегда будет найдено.

Заметим, что не все ветвления могут быть отмечены: так, использование динамических методов вызова приводит к тому, что до этапа выполнения невозможно определить, возникнет ли исключение. В других случаях применение этой технологии может привести к тому, что будет затрачено больше тактов, чем сэкономлено.

После этого компилятор транслирует исходный код в машинный и упаковывает команды в 128-битные пакеты. Шаблон пакета (bundle's template field) указывает не только на то, какие команды в пакете могут выполняться независимо, но и какие команды из следующего пакета могут выполняться параллельно. Команды в пакетах не обязательно должны быть расположены в том же порядке, что и в машинном коде, и могут принадлежать к различным путям ветвления. Компилятор может также помещать в один пакет зависимые и независимые команды, поскольку возможность параллельного выполнения определяется шаблоном пакета. В отличие от некоторых ранее существовавших архитектур со сверхдлинными словами команд (VLIW), IA-64 не добавляет команд "нет операции" (NOPS) для дополнения пакетов.

Во время выполнения программы IA-64 просматривает шаблоны, выбирает взаимно независимые команды и распределяет их по функциональным узлам. После этого производится распределение зависимых команд. Когда процессор обнаруживает "отмеченное" ветвление, вместо попытки предсказать значение условия ветвления и перехода к блоку, соответствующему предсказанному пути, процессор начинает параллельно выполнять блоки, соответствующие всем возможным путям ветвления. Таким образом, на машинном уровне ветвления нет.

Разумеется, в какой-то момент процессор наконец вычислит значение условия ветвления в нашем операторе IF-THEN-ELSE. Предположим, оно равно TRUE, следовательно, правильный путь отмечен предикатом P1. 6-битному полю предиката соответствует набор из 64 предикатных регистров (predicate registers) P0-P63 длиной 1 бит. Процессор записывает 1 в регистр P1 и 0 во все остальные.

К этому времени процессор, возможно, уже выполнил некоторое количество команд, соответствующих обоим возможным путям, но до сих пор не сохранил результат. Перед тем, как сделать это, процессор проверяет соответствующий предикатный регистр. Если в нём 1 — команда верна и процессор завершает её выполнение и сохраняет результат. Если 0 — результат сбрасывается.

Технология "отмеченных команд" существенно снижает негативное влияние ветвлений на машинном уровне. В то же время, если компилятор не "отметил" ветвление, IA-64 действует практически так же, как и современные процессоры: пытается предсказать путь ветвления и т.д. Испытания показали, что описанная технология позволяет устранить более половины ветвлений в типичной программе, и, следовательно, уменьшить более чем в два раза число возможных ошибок в предсказаниях.

Другой ключевой особенностью IA-64 является предварительная загрузка данных. Она позволяет не только загружать данные из памяти до того, как они понадобятся программе, но и генерировать исключение только в случае, если загрузка прошла неудачно. Цель предварительной загрузки — разделить собственно загрузку и использование данных, что позволяет избежать простоя процессора. Как и в технологии "отмеченных команд" здесь также сочетается оптимизация на этапе компиляции и на этапе выполнения.

Сначала компилятор просматривает код программы, определяя команды, использующие данные из памяти. Везде, где это возможно, добавляется команда предварительной загрузки на достаточно большом расстоянии перед командой, использующей данные и команда проверки загрузки непосредственно перед командой, использующей данные.

На этапе выполнения процессор сначала обнаруживает команду предварительной загрузки и, соответственно, пытается загрузить данные из памяти. Иногда попытка оказывается неудачной - например, команда, требующая данные, находится после ветвления, условия которого ещё не вычислены. "Обычный" процессор тут же генерирует исключение. IA-64 откладывает генерацию исключения до того момента, когда встретит соответствующую команду проверки загрузки. Но к этому времени условия ветвления, вызывавшего исключение, уже будут вычислены. Если команда, инициировавшая предварительную загрузку, относится к неверному пути, загрузка признается неудачной и генерируется исключение. Если же путь верен, то исключение вообще не генерируется. Таким образом, предварительная загрузка в архитектуре IA-64 работает аналогично структуре типа TRY-CATCH.

Возможность располагать команду предварительной загрузки до ветвления очень существенна, так как позволяет загружать данные задолго до момента использования (напомню, что в среднем каждая шестая команда является командой ветвления).

В 80-е годы некоторые разработчики RISC-процессоров высмеивали CISC-архитектуру и предрекали скорую погибель семейству x86. Но технологии и бизнес — разные вещи. Несмотря на технологические преимущества RISC-архитектуры, огромные ресурсы корпорации Intel и господство операционных систем DOS и Windows привели к тому, что процессоры архитектуры x86 остаются конкурентоспособными до сих пор. Теперь уже Intel заявляет, что RISC- архитектура устарела. Не совершает ли корпорация той же ошибки? В любом случае, до выхода в свет первого процессора архитектуры IA-64 остаётся ещё два года, и у конкурентов есть время принять ответные меры.