Министерство общего и профессионального образования РФ
Камышинский технологический институт
Филиал Волгоградского государственного технического университета

Реферат

по дисциплине:
Организация ЭВМ

Intel Pentium 4 3,06 ГГц с поддержкой технологии Hyper-Threading

Выполнили:
студент группы квт003
Вершков А.С

Камышин, 2002

План

1. Введение

Казалось бы не так уж и давно вышел Pentium 4 2,8 ГГц, но компания Intel видимо настолько горда способностью своего нового процессорного ядра к постоянному “разгону”, что не дает нам покоя анонсами все новых и новых процессоров. Однако сегодняшний процессор отличается от предыдущей топовой модели не только на 200 с небольшим мегагерц — то, о чем давно мечтали многие пользователи, наконец-то свершилось: технология эмуляции двух процессоров на одном процессорном ядре, ранее бывшая достоянием лишь сверхдорогих Xeon.
Все последующие модели Pentium 4, начиная с рассматриваемого, будут обладать поддержкой технологии Hyper-Threading. Однако кто-то может вполне резонно поинтересоваться: «А зачем мне двухпроцессорная машина дома? И действительно — зачем? Именно это я и постарался объяснить ниже. Итак: Hyper-Threading — что это такое и зачем он может быть нужен в обычных персональных компьютерах?

Intel Pentium 4 3,06 ГГц Socket 478, с поддержкой Hyper-Threading

2. *SMP и Hyper-Threading*

Как работает классическая SMP(Symmetric Multi-Processor) - система с точки зрения обычной логики? Не так уж велико количество пользователей, хорошо себе представляющих как работает SMP-система, и в каких случаях от использования двух процессоров вместо одного можно ожидать реального увеличения быстродействия, а в каких — нет.

Итак, представим, что у нас есть, к примеру, два процессора вместо одного. Что это дает?

В общем-то ничего. Потому что в дополнение к этому нужна еще и операционная система, умеющая задействовать эти два процессора. Эта система должна быть по определению многозадачной (иначе никакого смысла в наличии двух CPU просто быть не может), но кроме этого, ее ядро должно уметь распараллеливать вычисления на несколько CPU. Классическим примером многозадачной ОС, которая этого делать не умеет, являются все ОС от Microsoft, называемые обычно для краткости “Windows 9x” — 95, 95OSR2, 98, 98SE, Me. Они просто-напросто не могут определить наличие более чем одного процессора в системе.

Поддержкой SMP обладают ОС этого же производителя, построенные на ядре NT: Windows NT 4, Windows 2000, Windows XP. Также этой поддержкой обладают все ОС, основанные на идеологии Unix — всевозможные Free- Net- BSD, коммерческие Unix (такие как Solaris, HP-UX, AIX), и многочисленные разновидности Linux.

Если же два процессора все же определились системой, то дальнейший механизм их задействования в общем довольно прост. Если в данный момент времени исполняется одно приложение — то все ресурсы одного процессора будут отданы ему, второй же будет просто простаивать. Если приложений стало два — второе будет отдано на исполнение второму CPU, так что по идее скорость выполнения первого не должна уменьшиться. Однако на самом деле все сложнее.

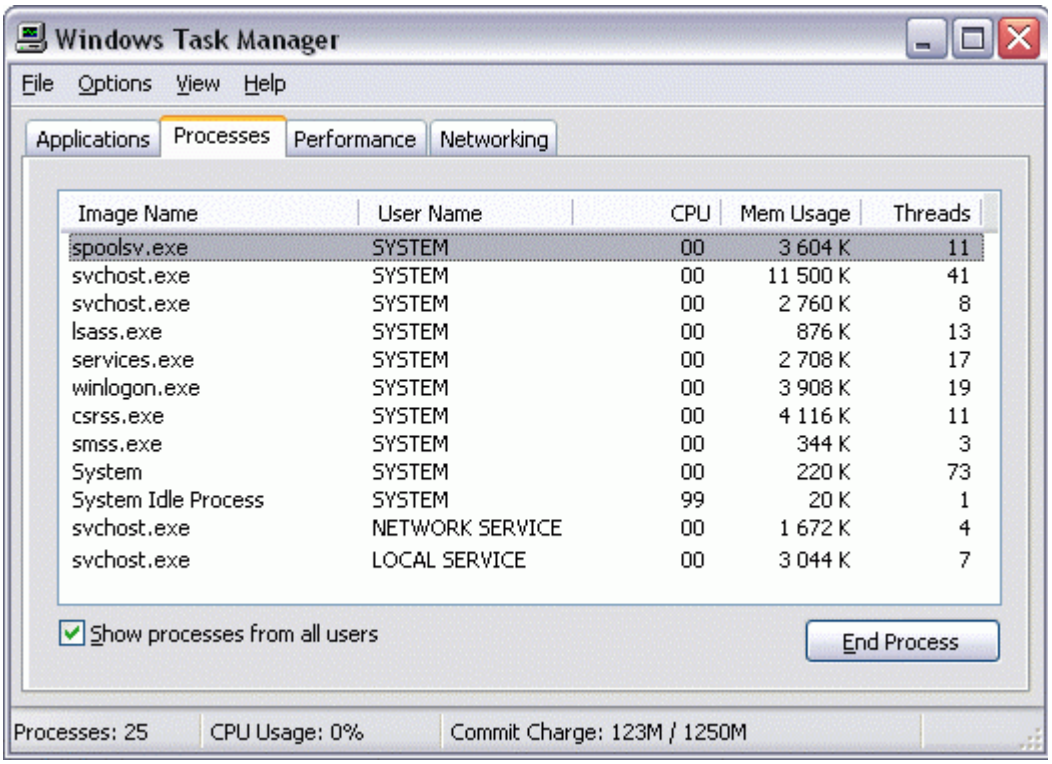
Исполняемое пользовательское приложение может быть запущено всего одно, но количество процессов (т. е. фрагментов машинного кода, предназначенных для выполнения некой задачи) в многозадачной ОС всегда намного больше. Поэтому на самом деле второй CPU способен немного “помочь” даже одиночной задаче, взяв на себя обслуживание процессов, порожденных операционной системой.

Кроме того, даже одно приложение может порождать потоки (threads), которые при наличии нескольких CPU могут исполняться на них по отдельности. Так, например, поступают почти все программы рендеринга — они специально писались с учетом возможности работы на многопроцессорных системах. Поэтому в случае использования потоков выигрыш от SMP иногда довольно весом даже в “однозадачной” ситуации.

По сути, поток отличается от процесса только двумя вещами — он во-первых никогда не порождается пользователем (процесс может запустить как система, так и человек, в последнем случае процесс = приложение; появление потока инициируется исключительно запущенным процессом), и во-вторых — поток выгружается вместе с родительским процессом независимо от своего желания.

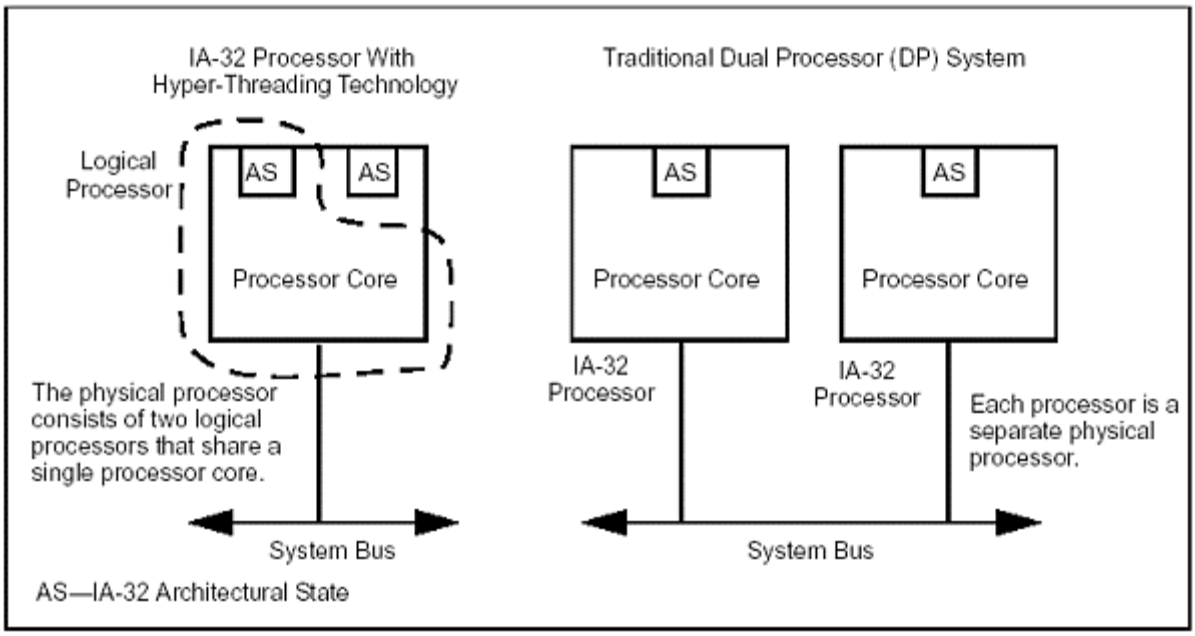
Также не стоит забывать, что в классической SMP-системе оба процессора работают каждый со своим кэшем и набором регистров, но память у них общая. Поэтому если две задачи одновременно работают с ОЗУ, мешать они друг другу будут все равно, даже если CPU у каждой свой.

Ну и наконец последнее: в реальности пользователь имеет дело не с одним, не с двумя, и даже не с тремя процессами. На приведенном коллаже (это действительно коллаж, потому что со скриншота Task Manager были удалены все пользовательские процессы, т. е. приложения, запускаемые “для работы”) хорошо видно, что “голая” Windows XP, сама по себе, не запустив еще ни одного приложения, уже породила 12 процессов, причем многие из них к тому же еще и многопоточные, и общее количество потоков достигает двухсот восьми штук!



Поэтому рассчитывать на то, что удастся прийти к схеме “по собственному CPU на каждую задачу” совершенно не приходится, и переключаться между фрагментами кода процессоры будут все равно — и физические, и виртуальные. Впрочем, на самом деле все не так грустно — при грамотно написанном коде ничего в данный момент не делающий процесс (или поток) процессорного времени практически не занимает (это тоже видно на коллаже).

Теперь, разобравшись с «физической» многопроцессорностью, перейдем к Hyper-Threading. Фактически — это тоже многопроцессорность, только виртуальная. Ибо процессор Pentium 4 на самом деле один. А процессоров ОС видит — два. Как это ?

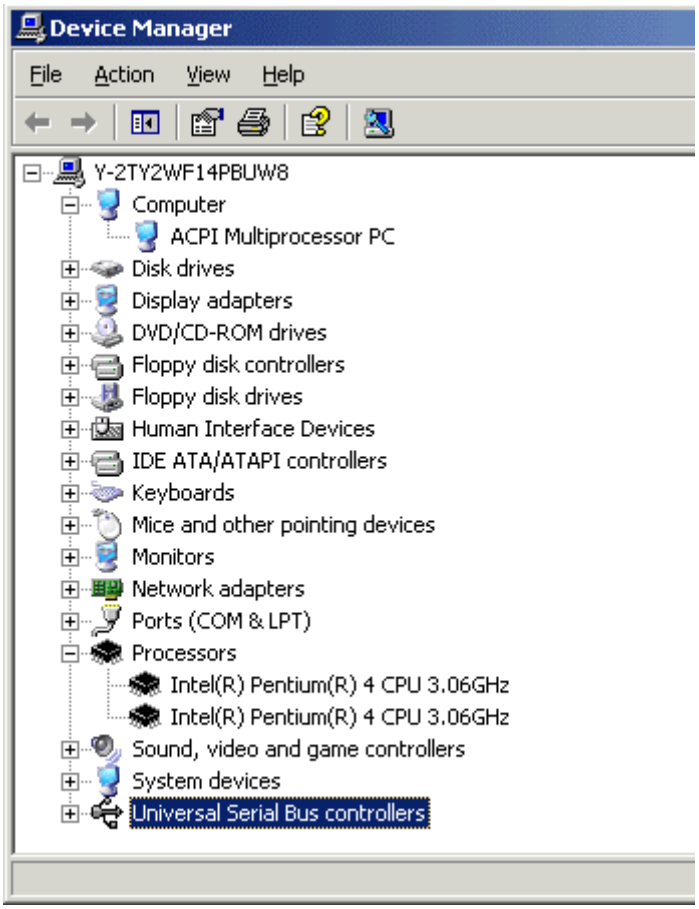


Классическому “одноядерному” процессору добавили еще один блок AS — IA-32 Architectural State. Architectural State содержит состояние регистров (общего назначения, управляющих, APIC, служебных). Фактически, AS#1 плюс единственное физическое ядро (блоки предсказания ветвлений, ALU, FPU, SIMD-блоки и пр.) представляет из себя один логический процессор (LP1),

а AS#2 плюс все то же физическое ядро — второй логический процессор (LP2). У каждого LP есть свой собственный контроллер прерываний (APIC — Advanced Programmable Interrupt Controller) и набор регистров. Для корректного использования регистров двумя LP существует специальная таблица — RAT (Register Alias Table), согласно данным в которой можно установить соответствие между регистрами общего назначения физического CPU. RAT у каждого LP своя. В результате получается схема, при которой на одном и том же ядре могут свободно выполняться два независимых фрагмента кода т. е. де-факто — многопроцессорную систему!

3. Hyper-Threading: совместимость

Кроме того, возвращаясь к вещам практическим и приземленным, хотелось бы затронуть еще один немаловажный аспект: *не все ОС, даже поддерживающие многопроцессорность, могут работать с таким CPU как с двумя*. Связано это с таким "тонким" моментом, как изначальное определение количества процессоров при инициализации операционной системы. Intel прямо заявляет, что ОС без поддержки ACPI второй логический процессор увидеть не смогут. Кроме того, BIOS системной платы также должен уметь определять наличие процессора с поддержкой Hyper-Threading. Фактически, применительно, к примеру, к Windows, это означает, что оказывается неприемлемой не только линейка Windows 9x, но и Windows NT — последняя ввиду отсутствия поддержки ACPI не сможет работать с одним новым Pentium 4 как с двумя.
Несмотря на заблокированную возможность работы с двумя физическими процессорами, с двумя логическими, получаемыми с помощью Hyper-Threading, сможет работать Windows XP Home Edition.



А Windows XP Professional, кстати, несмотря на ограничение количества физических процессоров до двух, при двух установленных CPU с поддержкой Hyper-Threading честно "видит" четыре.
К сожалению новые CPU с частотой более 3 ГГц могут потребовать замены системной платы.
Даже при номинальном сохранении все того же процессорного разъема Socket 478 Intel не удалось оставить в неприкосновенности потребляемую мощность и тепловыделение новых процессоров. Увеличение потребления по току связано не только с ростом частоты, но и с тем, что из-за ожидаемого использования "виртуальной многопроцессорности" нагрузка на ядро в среднем вырастет, следовательно, возрастет и средняя потребляемая мощность. "Старые" системные платы в некоторых случаях могут быть совместимы с новыми CPU — но только если делались "с запасом". Грубо говоря, те производители, которые делали свои платы в соответствии с рекомендациями самой Intel относительно потребляемой Pentium 4 мощности, оказались в проигрыше по отношению к тем, кто немного перестраховался.Но и это еще не все. Кроме ОС, BIOS и электроники платы, с технологией Hyper-Threading должен быть совместим еще *и* чипсет. Поэтому счастливыми обладателями двух процессоров по цене одного смогут стать

только те, чья системная плата основана на одном из новых чипсетов с поддержкой 533 МГц FSB: i850E, i845E, i845PE/GE.

4. Hyper-Threading: зачем она нужна

Компания Intel, если внимательно посмотреть, никогда не отличалась *абсолютным* совершенством своих продуктов, более того — вариации на те же темы от других производителей подчас получались гораздо более интересными и концептуально стройными. Однако, как оказалось, абсолютно все делать совершенным и не нужно — главное чтобы чип олицетворял собой какую-то идею, и идея эта приходилась очень вовремя и к месту. И еще — чтобы ее просто не было у других.

Так было с Pentium, когда Intel противопоставила весьма производительному в целочисленных операциях AMD Am5x86 мощный FPU. Так было с Pentium II, который получил широкую шину и быстрый кэш второго уровня, благодаря чему за ним так и не смогли угнаться все процессоры Socket 7. Так было и с Pentium 4, который противопоставил всем остальным наличие поддержки SSE2 и быстрый рост частоты — и тоже де-факто выиграл. Сейчас Intel предлагает Hyper-Threading.

Я думаю, что стоит задуматься — почему производитель, известный грамотностью своих инженеров (ни слова про маркетологов) и громадными суммами, которые он тратит на исследования, предлагает эту технологию.

Объявить Hyper-Threading “очередной маркетинговой штучкой”, конечно, проще простого. Однако не стоит забывать, что это *технология*, она требует исследований, денег на разработку, времени, сил. Не проще ли было нанять за меньшую сумму еще одну сотню PR-менеджеров или сделать еще десяток красивых рекламных роликов? Видимо, не проще. А значит, “что-то в этом есть”. Следует попытаться понять даже не то, что получилось в результате, а то, чем руководствовались разработчики IAG (Intel Architecture Group), когда принимали решение — разрабатывать “эту интересную мысль” дальше, или отложить на потом.

Как ни странно, для того чтобы понять как функционирует Hyper-Threading, вполне достаточно понимать как работает любая многозадачная операционная система. И действительно — ведь исполняет же каким-то образом *один* процессор сразу десятки задач? Этот “секрет” всем уже давно известен — на самом деле, конечно одновременно все равно выполняется только одна (на однопроцессорной системе) задача, просто переключение между кусками кода разных задач выполняется настолько быстро, что создается иллюзия одновременной работы большого количества приложений.

По сути, Hyper-Threading предлагает то же самое, но реализована аппаратно, внутри самого CPU. Есть некоторое количество различных исполняющих блоков (ALU, MMU, FPU, SIMD), и есть два “одновременно” исполняемых фрагмента кода. Специальный блок отслеживает, какие команды из каждого фрагмента необходимо выполнить в данный момент, после чего проверяет, загружены ли работой *все* исполняющие блоки процессора. Если один из них простаивает, *и именно он может исполнить эту команду* — ему она и передается. Естественно, существует и механизм принудительного посыла команды на выполнение — в противном случае один процесс мог бы захватить весь процессор (все исполняющие блоки) и исполнение второго участка кода (исполняемого на втором “виртуальном CPU”) было бы прервано. Данный механизм (пока) не является интеллектуальным т. е. не способен оперировать *различными* приоритетами, а просто чередует команды из двух разных цепочек в порядке живой очереди. Если, конечно, не возникает ситуации, когда команды одной цепочки по исполняющим блокам нигде не конкурируют с командами другой. В этом случае получается действительно на 100% параллельное исполнение двух фрагментов кода.

Самое очевидное следствие применения технологии Hyper-Threading — повышение коэффициента полезного действия процессора. Действительно — если одна из программ использует в основном целочисленную арифметику, а вторая — выполняет вычисления с плавающей точкой, то во время

исполнения первой FPU просто ничего не делает, а во время исполнения второй — наоборот, ничего не делает ALU. Казалось бы, на этом можно закончить.
Однако это идеальный (с точки зрения применения Hyper-Threading) вариант. Следует рассмотреть и другой: обе программы задействуют одни и те же блоки процессора. Понятно, что ускорить выполнение в данном случае довольно сложно — ибо физическое количество исполняющих блоков от "виртуализации" не изменилось. А вот не замедлится ли оно?
В случае с процессором без Hyper-Threading имеется просто "честное" поочередное выполнение двух программ на одном ядре с арбитром в виде операционной системы, и общее время их работы определяется:
временем выполнения кода программы №1
временем выполнения кода программы №2
временными издержками на переключение между фрагментами кода программ №1 и №2

В случае с Hyper-Threading схема становится немного другой:
время выполнения программы №1 на процессоре №1 (виртуальном)
время выполнения программы №2 на процессоре №2 (виртуальном)
время на переключение одного физического ядра (как набора требуемых обеим программам исполняющих блоков) между двумя эмулируемыми "виртуальными CPU"

Остается признать, что и тут Intel поступает вполне логично: конкурируют между собой по быстродействию только пункты за номером три, и если в первом случае действие выполняется программно-аппаратно (ОС управляет переключением между потоками, задействуя для этого функции процессора), то во втором случае имеется *полностью аппаратное решение* — процессор все делает сам. Теоретически, аппаратное решение всегда оказывается быстрее программного.

Также одним из серьезнейших неприятных моментов является то, что Pentium 4 приходится иметь дело с классическим x86-кодом, в котором активно используется прямое адресование ячеек и даже целых массивов, находящихся за пределами процессора — в ОЗУ. Да и вообще большинство обрабатываемых данных чаще всего находится там . Поэтому делить между собой виртуальные CPU будут не только регистры, но и общую для обоих процессорную шину, минуя которую данные в CPU попасть просто не могут.
Однако тут есть один тонкий момент: *на сегодняшний день "честные" двухпроцессорные системы на Pentium III и Xeon находятся в точно такой же ситуации*! Потому что шина AGTL+, доставшаяся в наследство всем сегодняшним процессорам Intel от знаменитого Pentium Pro (в дальнейшем ее лишь подвергали модификациям, но идеологию практически не трогали) — всего одна, сколько бы CPU ни было установлено в системе.
Отойти от этой схемы на x86 попробовала только AMD со своим Athlon MP — у AMD 760MP/760MPX от каждого процессора к северному мосту чипсета идет *отдельная* шина. Впрочем, даже в таком варианте проблема отодвигается не очень далеко — так как шина памяти точно одна — причем вот в этом случае уже везде.
Но и даже из этого в общем-то не очень приятного момента Hyper-Threading может помочь извлечь какую-то пользу. Дело в том, что по идее должен будет наблюдаться существенный прирост производительности не только в случае с несколькими задачами, использующими разные функциональные блоки процессора, но и в том случае, если задачи по-разному работают с данными, находящимися в ОЗУ. Если одно приложение что-то усиленно считает "внутри себя", другое же — постоянно подкачивает данные из ОЗУ, то общее время выполнения их в случае использования Hyper-Threading по идее должно уменьшиться даже если они используют одинаковые блоки исполнения инструкций — хотя бы потому, что команды на чтение данных из памяти смогут обрабатываться в то время, пока наше первое приложение будет что-то считать.

5. Заключение

В очередной раз, к радости всего прогрессивного человечества, Intel выпустила новый Pentium 4, производительность которого еще выше чем у предыдущего Pentium 4, и дело тут не только в лишних двухстах мегагерцах, а и в новой технологии под названием – Hyper-Threading
Технология Hyper-Threading с теоретической точки зрения выглядит весьма неплохо и соответствует реалиям сегодняшнего дня. Уже довольно редко можно застать пользователя с одним сиротливо открытым окном на экране — всем хочется одновременно и музыку слушать, и по Internet бродить, и диски с любимыми MP3 записывать, а может даже, и поиграть на этом фоне в какую-нибудь компьютерную игру.
Hyper-Threading позволяет увеличить коэффициент полезного действия процессора в определенных ситуациях. В частности — в ситуациях, когда одновременно исполняются разнородные по характеру приложения. Это конечно плюс, но он не является всеобъемлющим и глобальным. Потому что эффект от Hyper-Threading наблюдается исключительно в некоторых случаях. Понятно, что появление CPU, способного в два раза быстрее делать все то, что делалось ранее — это громадный прорыв. Однако Intel не стал инициировать начало новой эпохи перемен, просто добавив своему процессору возможность кое-что делать быстрее.
Однако Hyper-Threading нельзя назвать “бумажной” технологией, так как при определенных комбинациях она дает вполне ощутимый эффект. Даже намного больший эффект, чем иногда наблюдается при сравнении, к примеру, двух платформ с одним процессором на разных чипсетах. Хотя эффект этот наблюдается не всегда, и существенно зависит от *стиля* работы пользователя с компьютером. Причем именно здесь проявляется то что: *Hyper-Threading — это не SMP*.
“Классический SMP-стиль”, где пользователь рассчитывает на реакцию столь же классической “честной” многопроцессорной системы, здесь не даст желаемого результата.
“Стиль Hyper-Threading” — это сочетание процессов “развлекательных” или “служебных” с процессами “рабочими”. Пользователь не получит существенного ускорения от CPU с поддержкой этой технологии в большинстве классических многопроцессорных задач, или если по привычке будет запускать только одно приложение в один момент времени. Но он скорее всего получит *уменьшение времени исполнения многих фоновых задач*, исполняемых в качестве “довеска” к обычной работе. Фактически, Intel просто еще раз напомнила всем нам, что операционные системы, в которых мы работаем — *многозадачные*. И предложила способ ускорения — но не столько одного какого-то процесса самого по себе, сколько *комплекса выполняемых одновременно приложений*. Это интересный и достаточно востребованный подход.